NASA CR-178,410

NASA Contractor Report 178410

NASA-CR-178410
19880015235

USER'S GUIDE TO THE FAULT INFERRING NONLINEAR DETECTION SYSTEM (FINDS) COMPUTER PROGRAM

A.K. Caglayan, P.M. Godiwala, and H.S. Satz

CHARLES RIVER ANALYTICS INC.
Cambridge, MA

Contract NAS1-17719

June 1988

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665-5225

USER'S GUIDE TO THE FAULT INFERRING NONLINEAR DETECTION (FINDS) COMPUTER PROGRAM

TABLE OF CONTENTS

N88-24619 #

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

This report describes the operation and internal structure of the computer program FINDS (Fault Inferring Nonlinear Detection System) developed by Charles River Analytics Inc. for the NASA Langley Research Center. FINDS has been developed to provide detection, isolation, and compensation for hardware failures in the flight control sensors and ground-based navigation aids [1-4].

The FINDS algorithm is designed to provide reliable estimates for aircraft position, velocity, attitude, and horizontal winds to be used for guidance and control laws in the presence of possible failures in the avionics sensors. The FINDS algorithm exploits analytic redundancy between similar as well as dissimilar sensors; it can isolate a failure in a duplicate sensor configuration and detect a failure even if there is only one sensor of a given type in the configuration. FINDS can also detect simultaneous failures in navigation aid sensors, arising for instance from ground antenna malfunctions. Hence, FINDS can be used to increase the reliability of a sensor configuration with a given redundancy. For example, the fail-operational/fail-safe capability of a triply redundant voting system can be improved to at least a fail-op/fail-op/fail-safe capability. Conversely, FINDS can be employed to reduce the hardware redundancy requirements for a given reliability figure. As an example, FINDS can be used to replace a triply redundant voting system with dual redundancy while maintaining the overall reliability of the system.

The FINDS algorithm consists of 1) a no-fail filter (NFF), which is an extended Kalman filter (EKF) based on the assumption of no sensor failures and which provides estimates for aircraft states, horizontal winds, and normal operating sensor biases; 2) a set of test-of-mean detection tests implemented over moving windows of the NFF residuals; 3) a bank of first order filters activated upon failure detection to estimate failure levels in individual

sensors; and 4) a decision function which isolates the failed sensor by selecting the most likely failure mode depending on the likelihood ratios. When a sensor failure is detected and isolated, the algorithm is restructured to eliminate the failed sensor from further processing and to remove the accumulated effects of the sensor failure on the NFF. Failure identification decisions are monitored with the use of a healer algorithm; sensors falsely identified as failed or sensors recovered from failures are restored to the system.

The FINDS algorithm was developed with the use of a digital simulation of a commercial transport aircraft (B-737) [1-4]. Flight recorded data for this aircraft were used to address the issues of sensor modeling inaccuracies, such as time-varying sensor bias and time correlated noise [5-6]. The FINDS algorithm was then modified to "fit" the size constraints of a flight computer and to meet real-time execution requirements without compromising sensor failure detection and isolation (FDI) and state estimation performance [7-10].

To meet the real-time execution requirements, the FINDS algorithm has been partitioned to execute on a dual parallel processor configuration: one based on the translational dynamics and the other on the rotational kinematics. In addition, a new hierarchical failure isolation strategy has been developed, replacing the multiple hypothesis test in the earlier versions. Finally, a multi-rate implementation of the FINDS algorithm has been implemented to further increase execution speed.

The outline of the report is as follows. An overview of the FINDS algorithm is given in the next section. The implemented equations are given in detail in Section 3. Section 4 contains the flow charts for the key subprograms. The input and output files are discussed in Section 5. Program variable indexing convention is presented as tables in Section 6. Subprogram descriptions are presented in Section 7. Finally, Section 8 contains the common block descriptions used in the program.

2. FINDS ALGORITHM OVERVIEW

Given a configuration of avionics sensors on an aircraft, the FINDS algorithm generates fault tolerant estimates for the vehicle states as required by the flight control, guidance, and navigation systems in the presence of possible sensor failures. The desired qualities of FINDS are 1) use of analytical redundancy concepts to minimize hardware replication requirements; 2) timely detection of sensor failures; 3) ability to detect all types of sensor failures; 4) acceptable false alarm/detection probability performance; 5) ability to recover from false alarms; and 6) minimal computational complexity to permit real time operation on flight qualified computers.

The FINDS algorithm baseline structure is shown in Figure 2.1. The replicated sensor measurements are separated according to their function in the no-fail filter. That is, accelerometer and gyro measurements are used as input sensors to integrate the vehicle point mass equations of motion, and the remaining sensors (MLS, IAS, and IMU) are used as measurement sensors. The input sensors are processed in selection logic, and similar measurement sensors are averaged to reduce the overall complexity of the computations without a loss of generality.

The NFF shown in Figure 2.1 is an EKF which is implemented on the assumption of no sensor failures. The EKF development is based on discrete time difference equations for the vehicle equations of motion. The NFF provides estimates for the aircraft position, velocity, attitude, and horizontal winds, and estimates for the normal operating biases associated with a specified subset of the input and measurement sensors.

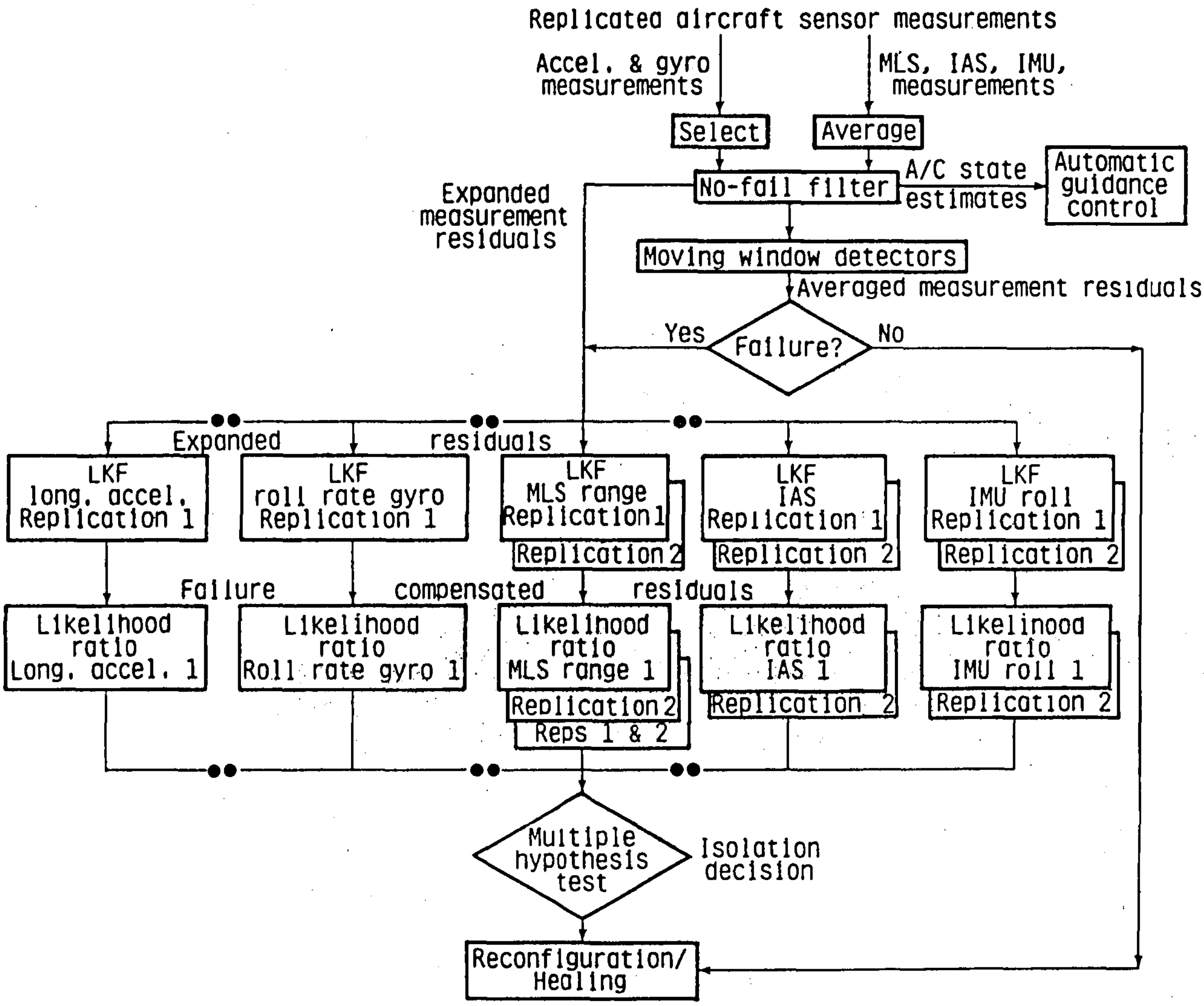

Figure 2.1: FINDS Algorithm Baseline Structure

The formulation yields a computationally efficient EKF implementation in which the input sensors are integrated into the NFF without closed loop filtering. Only one set of input sensors and the average of the measurement sensors are used. The remaining replicated sensors are held in standby and inserted as failures are detected and isolated. A decomposition procedure based on the separated EKF algorithm provides the EKF filter gains [11]-[12].

The NFF also generates a residual sequence for the averaged measurements, as seen in Figure 2.1, and a detection test is performed on these residuals over a moving window. The length of the moving window is different for input sensors and measurement sensors. A test of mean is compared to a predetermined threshold to determine a sensor failure. If a sensor failure is detected, the bank of detectors is run using the saved residuals in the corresponding moving window memory. The failure levels are estimated and the failure is isolated depending on the computed likelihood ratios.

When a failure is isolated, a reconfiguration algorithm is used to restructure the FINDS algorithm [13]. When a gyro or accelerometer (input sensor) fails, the faulty sensor is replaced. If there are no more valid sensors of that type, the NFF is restructured, provided it is able to function with the remaining set of sensors. When a measurement sensor fails, the isolated sensor is flagged to be inactive, and appropriate changes are made in the NFF noise statistics; also, the NFF is collapsed to accommodate the loss of all the sensors of a given type. The reconfiguration block also functions to reinitialize the NFF, detectors, and likelihood ratios following identification of a failure.

To recover from false alarms, each failed sensor is given a healing test. Input sensors are tested by comparison with sensors of the same type used by the NFF. A failed measurement sensor is tested with the NFF estimate of that

sensor. These are binary hypothesis tests conditioned on the decision rule that the sensor currently in use is healthy.

The NFF state estimates are initialized using the first iteration of the flight data, which includes MLS azimuth, elevation, and range, IAS, and IMU pitch, roll, and yaw measurements, to compute the aircraft position, velocity, attitude, and horizontal winds in the runway frame, shown in Figure 2.2 as required by the NFF.

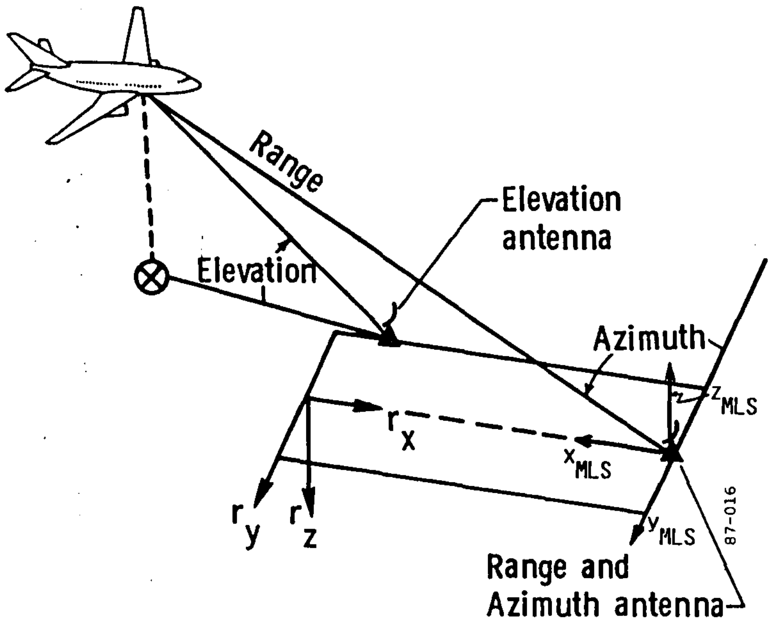

Figure 2.2: Runway Coordinate System and MLS Geometry

3. FINDS ALGORITHM IMPLEMENTATION

The interactive version of FINDS suitable for operation in a simulation environment was developed on a DEC VAX 11/780 using FORTRAN 77 under the VMS operating system. The flight data driven version of FINDS suitable for operation using either flight recorded or simulation generated sensor data was developed on Charles River Data Systems Universe 68/35 using FORTRAN 77 under the UNOS operating system, and SUN 3/160 using FORTRAN 77 under SunOS operating system. Several modifications have been made to the interactive version of FINDS to reduce the size and increase the speed of the algorithm, and to improve state estimation and sensor FDI performance. The composite version, FINDSCMP, of FINDS incorporates these changes, in particular, the hierarchical isolation strategy and multi-rate implementation. In addition, the FINDS algorithm has been partitioned into two parts for a parallel processing architecture: FINDS1 processing the sensors related to rotational kinematics and FINDS2 processing the sensors related to the translational dynamics. The partitioned version of FINDS has been ported onto a dual-processor configured ROLM 1666 flight computer using ROLM FORTRAN 66 compiler under the ROLM Real Time Operating System. A DMA local data communication link has been used for communication among the processors. In this section, the implemented equations for FINDSCMP, FINDS1, and FINDS2 are described. The execution flow of the main program is illustrated in Figure 3.1.

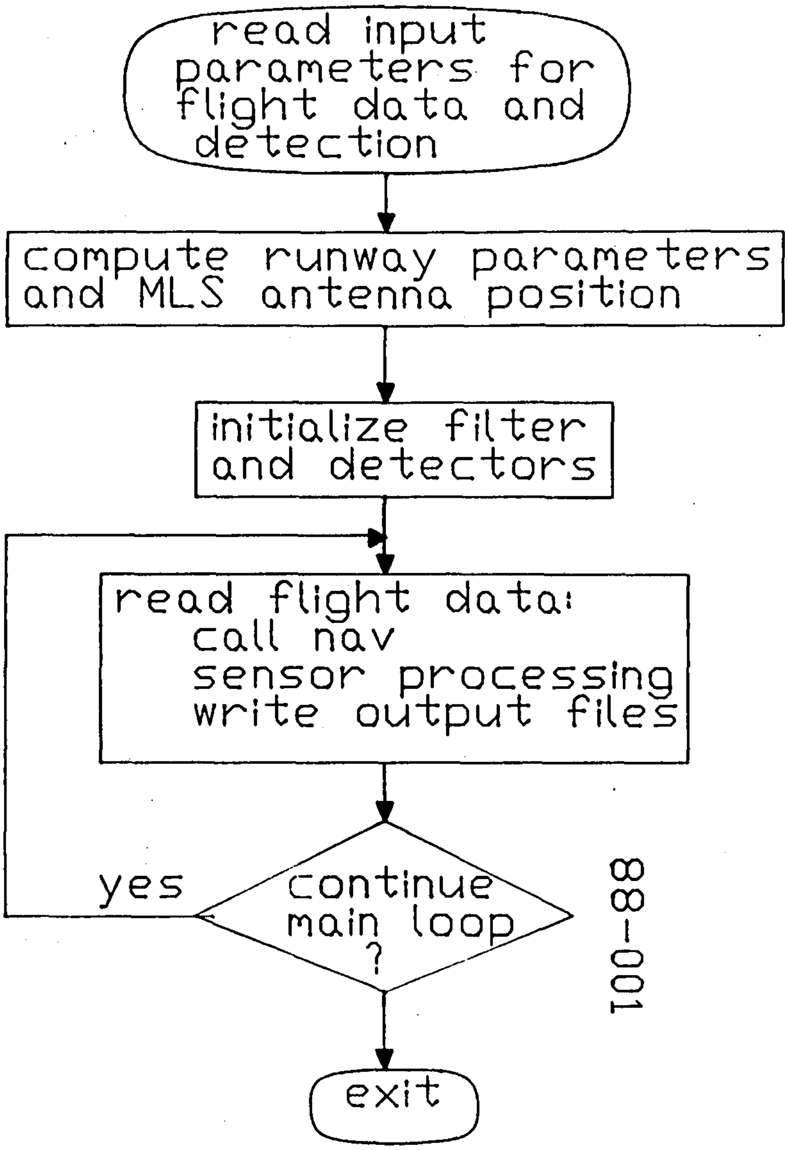

Figure 3.1: FINDS Main Program Execution Flow

FINDSCMP (Composite Version)

Number of states, NX = 11

State vector, $\hat{x} = [\hat{r}_x, \hat{r}_y, \hat{r}_z, \hat{\dot{r}}_x, \hat{\dot{r}}_y, \hat{\dot{r}}_z, \hat{\phi}, \hat{\theta}, \hat{\psi}, \hat{w}_x, \hat{w}_y]^T$

Number of biases, NB = 6

Bias vector, $\hat{b} = [\hat{b}_{ax}, \hat{b}_{ay}, \hat{b}_{az}, \hat{b}_p, \hat{b}_q, \hat{b}_r]^T$

Number of measurement types, NY = 7

Measurement vector, $y = [MLS_{az}, MLS_{e\ell}, MLS_{rn}, IAS, IMU_{\phi}, IMU_{\theta}, IMU_{\psi}]$

Number of input types, NU1 = 6

Input vector, $u = [ax, ay, az, p, q, r]^T$

--- New Time Iteration Start: time `k` ----

READFL : read the NFF input sensors $u_i^n(k)$, and the NFF measurement sensors , $y_j^n(k)$; i=1,6 ; j=1,7 ; n=1,2 (dual replication)

INITXF: Compute the NFF initial state estimates using the first iteration of flight data. Denoting the aircraft position in the MLS frame by r_{xm}, r_{ym}, r_{zm}:

$$\hat{r}_{xm}(k_o) = \sqrt{f + (f^2 - h)}$$

$$\hat{r}_{ym}(k_o) = -y_{rn}(k_o)\cdot[\sin(y_{az}(k_o))]$$

$$\hat{r}_{zm}(k_o) = \sqrt{y_{rn}^2(k_o) - \hat{r}_{xm}^2(k_o) - \hat{r}_{ym}^2(k_o)}$$

where

$$f = x_{oe}\cdot [\sin(y_{e\ell}(k_o)]^2$$

$$h = (x_{oe}^2 + y_{oe}^2)\cdot[\sin(y_{e\ell}(k_o))]^2 + y_M^2 - y_{rn}^2(k_o)\cdot[\cos(y_{e\ell}(k_o))]^2$$
$$- 2\cdot y_M\cdot y_{oe}\cdot[\sin(y_{e\ell}(k_o))]^2 - (z_{oe}^2 - 2\cdot z_M\cdot z_{oe})\cdot[\cos(y_{e\ell}(k_o)]^2$$

where (x_{oe}, y_{oe}, z_{oe}) represent the coodinates of the elevation antenna in the MLS frame.

$$\hat{r}_x(k_o) = x_M - \hat{r}_{xm}(k_o)$$
$$\hat{r}_y(k_o) = y_M + \hat{r}_{ym}(k_o)$$
$$\hat{r}_z(k_o) = z_M - \hat{r}_{zm}(k_o)$$

where (x_M, y_M, z_M) are the azimuth/range antenna coordinates in the runway frame.

$$\hat{\dot{r}}_x(k_o) = y_{sp}(k_o)\cdot\cos(y_\theta(k_o))\cdot\cos(y_\psi(k_o)) + \hat{w}_x(k_o)$$
$$\hat{\dot{r}}_y(k_o) = y_{sp}(k_o)\cdot\cos(y_\theta(k_o))\cdot\sin(y_\psi(k_o)) + \hat{w}_y(k_o)$$
$$\hat{\dot{r}}_z(k_o) = -y_{sp}(k_o)\cdot\sin(y_\theta(k_o))$$
$$\hat{w}_x(k_o) = 0$$
$$\hat{w}_y(k_o) = 0$$

The initial estimates for the aircraft attitude are obtained by averaging the replicated IMU measurements:

$$\hat{\phi}(k_o) = (y^1_\phi(k_o) + y^2_\phi(k_o))/2$$
$$\hat{\theta}(k_o) = (y^1_\theta(k_o) + y^2_\theta(k_o))/2$$
$$\hat{\psi}(k_o) = (y^1_\psi(k_o) + y^2_\psi(k_o))/2 - \psi_R$$

where ψ_R is the runway yaw, fixed for the given runway configuration.

<u>SUMIN</u> : (i) compensate rate-gyros for earth's rotation effects

(ii) average inputs and compensate for biases:

$$\bar{u}_i(k) = \frac{u^c_i(k) + u^c_i(k-1)}{2} - \hat{b}_i(k-1)$$

where c denotes the current active replication

<u>EKFN1(2)</u>: (i) UPDB ---→ update input transition matrix $B(\hat{x}(k-1))$

$$B(\hat{x}(k-1))=\begin{bmatrix} \Delta^2/2 & T_{GB}(\hat{x}(k-1)) & 0 \\ \Delta & T_{GB}(\hat{x}(k-1)) & 0 \\ 0 & & \Delta\, T_{ER}(\hat{x}(k-1)) \\ 0 & & 0 \end{bmatrix}$$

where the transformation from the body axes into the ground frame is computed according to:

$$T_{GB}(\hat{x}(k-1)) = \begin{bmatrix} c\hat{\theta}c\hat{\psi} & s\hat{\phi}s\hat{\theta}c\hat{\psi}-c\hat{\phi}s\hat{\psi} & c\hat{\phi}s\hat{\theta}c\hat{\psi}+s\hat{\phi}s\hat{\psi} \\ c\hat{\theta}s\hat{\psi} & s\hat{\phi}s\hat{\theta}s\hat{\psi}+c\hat{\phi}c\hat{\psi} & c\hat{\phi}s\hat{\theta}s\hat{\psi}-s\hat{\phi}c\hat{\psi} \\ -s\hat{\theta} & s\hat{\phi}c\hat{\theta} & c\hat{\phi}c\hat{\theta} \end{bmatrix}$$

where $\hat{\phi}(k-1)$, $\hat{\theta}(k-1)$, $\hat{\psi}(k-1)$ are the NFF estimates for the Euler angles and c,s, and t are abbreviations for the cosine, sine and tangent functions, respectively. The matrix T_{ER} relating the body rates to the Euler angles is computed according to:

$$T_{ER}(\hat{x}(k))=\begin{bmatrix} 1 & t(\hat{\theta}(k))s(\hat{\phi}(k)) & t(\hat{\theta}(k))c(\hat{\phi}(k)) \\ 0 & c(\hat{\phi}(k)) & -s(\hat{\phi}(k)) \\ 0 & s(\hat{\phi}(k))sc(\hat{\theta}(k)) & c(\hat{\phi}(k))sc(\hat{\theta}(k)) \end{bmatrix}$$

where sc is the abbreviation for the secant function.

(ii) UPDQ $\longrightarrow$ update process noise covariance $Q(\hat{x}(k-1))$

$$Q(\hat{x}(k))=\begin{bmatrix} \frac{\Delta^3}{3}\, T_{GB}V_aT_{GB}^T & \frac{\Delta^2}{2}\, T_{GB}V_aT_{GB}^T & 0 & 0 \\ \frac{\Delta^2}{2}\, T_{GB}V_aT_{GB}^T & \Delta\, T_{GB}V_aT_{GB}^T & 0 & 0 \\ 0 & 0 & \Delta\, T_{ER}V_{rg}T_{ER}^T & 0 \\ 0 & 0 & 0 & \int_0^\Delta e^{A_w s}Q_w e^{A_w^T s}ds \end{bmatrix}$$

where V_a is the covariance for the accelerometer sensor noises given by

$$V_a = \begin{bmatrix} \sigma_{ax}^2 & 0 & 0 \\ 0 & \sigma_{ay}^2 & 0 \\ 0 & 0 & \sigma_{az}^2 \end{bmatrix}$$

where σ_{ax}, σ_{ay}, σ_{az} are the accelerometer sensor noise standard deviations. V_{rg} is the covariance for the rate gyro sensor noises given by:

$$V_{rg} = \begin{bmatrix} V_{rg}(1) & 0 & 0 \\ 0 & V_{rg}(2) & 0 \\ 0 & 0 & V_{rg}(3) \end{bmatrix}$$

with

$$V_{rg}(1) = \sigma_p^2 + SPM * (AQ^2 + AR^2) + SCF * AP^2$$
$$V_{rg}(2) = \sigma_q^2 + SPM * (AP^2 + AR^2) + SCF * AQ^2$$
$$V_{rg}(3) = \sigma_r^2 + SPM * (AP^2 + AQ^2) + SCF * AR^2$$

where σ_p, σ_q, σ_r are the rate gyro measurement noise standard deviations; AP, AQ, AR are the averaged p, q, r measurements passed through symmetric limiters with thresholds 4 deg/s, 1 deg/sec, and 2.5 deg/sec:

$$AP = \frac{pm^1(k) + pm^2(k)}{2} \; ; \quad AQ = \frac{qm^1(k) + qm^2(k)}{2} \; ; \quad AR = \frac{rm^1(k) + rm^2(k)}{2}$$

where SCF in the rate gyro scale factor error variance, and SPM is the sum of SCF and rate gyro misalignment error variances.

The wind model system matrix A_w is given by

$$A_W = \begin{bmatrix} -\frac{1}{\tau_w} & 0 \\ 0 & -\frac{1}{\tau_w} \end{bmatrix}$$

where τ_w is the time constant associated with the wind model.

(iii) Compute prediction error covariance via:

$$P_o(k/k-1) = A * P_o(k-1/k-1) * A^T + Q(\hat{x}(k-1))$$

where A is constant state transition matrix given by

$$A = \begin{bmatrix} I & \Delta I & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & e^{A_w \Delta} \end{bmatrix}$$

<u>BLEND(2)</u>: (i) Compute single stage prediction:

$$\hat{x}(k/k-1) = A * \hat{x}(k-1) + B(\hat{x}(k-1)) * \bar{u}(k)$$

(ii) Update single stage prediction for measurements UPDH --→ $h(\hat{x}(k/k-1))$

$$h_1(\hat{x}(k/k-1)) = \hat{y}_{az}(k/k-1) = \frac{1}{\sigma_{az}} [\sin^{-1}[(-\hat{r}_y(k/k-1) + y_M)/\hat{r}_{az}(k/k-1)]$$

$$h_2(\hat{x}(k/k-1)) = \hat{y}_{e\ell}(k/k-1) = \frac{1}{\sigma_{e\ell}} [\sin^{-1}[(-\hat{r}_z(k/k-1) + z_E)/\hat{r}_{e\ell}(k/k-1)]$$

$$h_3(\hat{x}(k/k-1)) = \hat{y}_{rn}(k/k-1) = \frac{1}{\sigma_{rn}}[\hat{r}_{az}(k/k-1)]$$

where (x_M, y_M, z_M) and (x_E, y_E, z_E) are the azimuth and elevation antenna positions in the runway frame, σ_{az}, $\sigma_{e\ell}$, and σ_{rn} are the averaged MLS sensor noise standard deviations, and $\hat{r}_{az}, \hat{r}_{e\ell}$ are single stage predictions for the aircraft range from the azimuth and elevation antennas given by:

$$\hat{r}_{az}(k/k-1) = \sqrt{(\hat{r}_x(k/k-1)-x_M)^2+(\hat{r}_y(k/k-1)-y_M)^2+(\hat{r}_z(k/k-1)-z_M)^2}$$

$$\hat{r}_{e\ell}(k/k-1) = \sqrt{(\hat{r}_x(k/k-1)-x_E)^2+(\hat{r}_y(k/k-1)-y_E)^2+(\hat{r}_z(k/k-1)-z_E)^2}$$

$$h_4(\hat{x}(k/k-1)) = \hat{y}_{sp}(k/k-1) = \frac{1}{\sigma_{sp}} \sqrt{[\hat{\dot{r}}_x(k/k-1)-\hat{w}_x(k/k-1)]^2+[\hat{\dot{r}}_y-\hat{w}_y(k/k-1)]^2+\hat{\dot{r}}_z^{\,2}}$$

where σ_{sp} is the averaged IAS sensor noise standard deviation.

$$h_5(\hat{x}(k/k-1)) = \hat{y}_\phi(k/k-1) = \frac{1}{\sigma_\phi} \hat{\phi}(k/k-1)$$

$$h_6(\hat{x}(k/k-1)) = y_\theta(k/k-1) = \frac{1}{\sigma_\theta}\hat{\theta}(k/k-1)$$

$$h_7(\hat{x}(k/k-1)) = y_\psi(k/k-1) = \frac{1}{\sigma_\psi}\hat{\psi}(k/k-1)$$

where σ_ϕ, σ_θ, σ_ψ are the averaged IMU sensor noise standard deviations.

<u>SUMOUT</u> : $$\bar{y}_j(k) = \frac{y_j^1(k) + y_j^2(k)}{2}$$

<u>EKFN1(1)</u>: (i) UPDPH --→ update the partials of the measurements

The nonzero elements of the measurement partial $H(\hat{x}(k/k-1))$ are computed according to:

$$H_{1,1} = \frac{\hat{r}_x(k/k-1)-x_M}{\hat{r}_{az}(k/k-1)\cdot\sigma_{az}}$$

$$H_{1,2} = \frac{\hat{r}_y(k/k-1)-y_M}{\hat{r}_{az}(k/k-1)\cdot\sigma_{az}}$$

$$H_{1,3} = \frac{\hat{r}_z(k/k-1)-z_M}{\hat{r}_{az}(k/k-1)\cdot\sigma_{az}}$$

$$H_{2,1} = \frac{(\hat{r}_x(k/k-1)-x_M)\ (\hat{r}_y(k/k-1)-y_M)}{\hat{r}^2_{az}(k/k-1)\cdot\hat{r}_{xz}(k/k-1)\cdot\sigma_{e\ell}}$$

where $\hat{r}_{xz}(k/k-1) = \sqrt{(\hat{r}_x(k/k-1)-x_M)^2 + (\hat{r}_z(k/k-1)-z_M)^2}$

$$H_{2,2} = \frac{-\hat{r}_{xz}(k/k-1)}{\hat{r}^2_{az}(k/k-1)\cdot\sigma_{e\ell}}$$

$$H_{2,3} = \frac{(\hat{r}_y(k/k-1)-y_M)\ (\hat{r}_z(k/k-1)-z_M)}{\hat{r}^2_{az}(k/k-1)\cdot\hat{r}_{xz}(k/k-1)\cdot\sigma_{e\ell}}$$

$$H_{3,1} = \frac{(\hat{r}_x(k/k-1)-y_E)\ (\hat{r}_z(k/k-1)-z_E)}{\hat{r}^2_{e\ell}(k/k-1)\cdot\hat{r}_{xy}(k/k-1)\cdot\sigma_{rn}}$$

where $\hat{r}_{xy}(k/k-1) = \sqrt{(\hat{r}_x(k/k-1)\ -x_E)^2 + (\hat{r}_y(k/k-1)-y_E)^2}$

$$H_{3,2} = \frac{(\hat{r}_y(k/k-1)-y_E)\ (\hat{r}_z(k/k-1)-z_E)}{\hat{r}^2_{e\ell}(k/k-1)\cdot\hat{r}_{xy}(k/k-1)\cdot\sigma_{rn}}$$

$$H_{3,3} = \frac{-\hat{r}_{xy}(k/k-1)}{\hat{r}^2_{e\ell}(k/k-1)\cdot\sigma_{rn}}$$

$$H_{4,4} = \frac{\hat{\dot{r}}_x(k/k-1)-\hat{w}_x(k/k-1)}{\hat{s}(k/k-1)\cdot\sigma_{sp}}$$

where $\hat{s}(k/k-1) = \sqrt{(\hat{\dot{r}}_x(k/k-1)\ -\hat{w}_x)^2 + (\hat{\dot{r}}_y(k/k-1)-\hat{w}_y)^2 + \hat{\dot{r}}_z(k/k-1)}$

$$H_{4,5} = \frac{\hat{\dot{r}}_y(k/k-1)-\hat{w}_y(k/k-1)}{\hat{s}(k/k-1)\cdot\sigma_{sp}}$$

$$H_{4,6} = \frac{\hat{\dot{r}}_z(k/k-1)}{\hat{s}(k/k-1)\cdot\sigma_{sp}}$$

$$H_{4,10} = -H(4,4)$$

$$H_{4,11} = -H(4,5)$$

(ii) Compute the bias-free NFF gain:

$$K_x(k) = P_o(k/k-1) * [H * P_o(k/k-1) * H^T + R(k)]^{-1}$$

(iii) Compute the bias-free NFF single stage prediction error covariance:

$$P_o(k/k) = [I - K_x * H] * P_o(k/k-1) * [I - K_x * H]^T +$$
$$K_x * R(k) * K_x^T$$

where $R(k) = \text{diag}\ \{1/c_i\}$

<u>BIASF(1)</u>: (i) Update bias observation matrix:

$$C_b(k) = H * [A * V_b(k-1) + B] + D$$

(ii) Update bias propagation matrix:

$$V_b(k) = [I - K_x * H] * A * V_b(k-1) + [-B + K_x * (H * B - D)]$$

(iii) Compute the NFF bias gain:

$$K_b(k) = P_b(k-1) * C_b^T(k) + [C_b(k) * P_b(k-1) * C_b^T(k) + R_b(k)]^{-1}$$

where $R_b(k) = [H * P_o(k/k-1) * H^T + R(k)]$ from EKFN1(1)

(iv) Compute the NFF bias estimation error covariance:

$$P_b(k) = [I - K_b(k) * C_b(k)] * P_b(k-1)$$

<u>BLEND(1)</u>: (i) Compute averaged measurement residuals:

$$r(k) = \bar{y}(k) - h(\hat{x}(k/k-1))$$

(ii) Update state estimate:

$$\hat{x}(k) = \hat{x}(k/k-1) + [K_x(k) + V_b(k) * K_b(k)] * r(k)$$

(iii) Update bias estimates:

$$\hat{b}(k) = \hat{b}(k-1) + K_b(k) * r(k)$$

<u>DESCMP</u>: Evaluate expanded measurement residual and store in moving window

$$r(k) = \begin{bmatrix} y_i^1/\sigma_i - h_i(\hat{x}(k/k-1)) \\ y_i^2/\sigma_i - h_i(\hat{x}(k/k-1)) \end{bmatrix}$$

<u>DET01</u>: (i) Compensate measurement residual covariance inverse RTINV using sensor noise parameters for window 01

(ii) Compute the likelihood ratio for moving window 01

$$LRT01(K) = r^T(k) * RTINV_{01} * r(k)$$

If LRT01 < $threshold_{01}$, then no measurement sensor failure

else ISOLATE (01)

DET05: (i) Compensate RTINV for moving window 05

(ii) Compute measurement residual average for moving window 05:

$$\bar{r}_{05}(k) = \frac{1}{5} \sum_{j=k-4}^{k} r(j)$$

(iii) Compute likelihood ratio LRT05 for under 05

$$LRT05(k) = \bar{r}^T_{05}(k) * RTINV_{05} * \bar{r}_{05}(k)$$

If LTR05 < $threshold_{05}$, then no sensor failures

else ISOLATE (05)

DET10: (i) Compensate RTINV for window 10

(ii) Compute measurement residual average for moving window 10

$$\bar{r}_{10}(k) = \frac{1}{10} \sum_{j=k-9}^{k} r(j)$$

(iii) Compute likelihood ratio LRT10 for moving window 10

$$LRT10(k) = \bar{r}^T_{10}(k) * RTINV_{10} * \bar{r}_{10}(k)$$

If LTR10 < $threshold_{10}$, then no sensor failures

else ISOLATE (10)

ISOLAT (w): (i) Form prediction error covariance for composite state:

$$PXF(k) = \begin{bmatrix} P_x(k) & P_{xb}(k) \\ P^T_{xb}(k) & P_b(k) \end{bmatrix}$$

where

$$P_x(k) = P_o(k/k) + [A * VB(k) + B(\hat{x}(k-1)] * P_b(k/k) * [A * VB(k) + B(\hat{x}(k-1)\ ]^T$$

$$P_{xb}(k) = [A * VB(k) + B(\hat{x}(k-1)]\ P_b(k/k)$$

$$P_b(k) = P_b(k/k)$$

(ii) Compute inverse of innovation covariance

$$\tilde{R}^{-1}(k) = \left\{[\bar{H}\ \bar{D}] * PXF(k) * [\bar{H}\ \bar{D}]^T + \begin{bmatrix} R & 0 \\ 0 & R \end{bmatrix}\right\}^{-1}$$

$$R = \text{diag}\ \left[(\sigma_{dw}/\sigma_i)**2\right]$$

(iii) Compute failure observation matrix:

$$C_i(k,\hat{x}(k)) = [\bar{H}\ \bar{D}] \begin{bmatrix} A & -B(\hat{x}(k-1)) \\ 0 & I \end{bmatrix} * \begin{bmatrix} V_{ix}(k-1) \\ V_{ib}(k-1) \end{bmatrix} +$$

$$[\bar{H}\ \bar{D}] * \begin{bmatrix} -B_i(\hat{x}(k-1)) \\ 0 \end{bmatrix} + D_i(1/\sigma_i)$$

(iv) Compute failure propagation matrix

$$\begin{bmatrix} V_{ix}(k) \\ V_{ib}(k) \end{bmatrix} = \left\{\begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \quad \begin{bmatrix} K_o(k) \\ K_b(k) \end{bmatrix} \quad [\bar{H}\quad \bar{D}]\right\} *$$

$$\begin{bmatrix} A & -B(\hat{x}(k-1)) \\ 0 & I \end{bmatrix} \begin{bmatrix} V_{ix}(k-1) \\ V_{ib}(k-1) \end{bmatrix} + \begin{bmatrix} -B_i(\hat{x}(k-1)) \\ 0 \end{bmatrix}$$

$$- \begin{bmatrix} K_o(k) \\ K^o(k) \end{bmatrix} \left\{[\bar{H}\quad \bar{D}] \begin{bmatrix} -B_i(\hat{x}(k-1)) \\ 0 \end{bmatrix} + D_i(1/\sigma_i)\right\}$$

(v) Compute failure level estimates $\hat{m}_i(k) = \hat{m}_i(k-1) + G_i(k) * RES(k)$

where

$$RES(k) = [r_o(k) - C_i(k) * \hat{m}_i(k-1)]$$

$$G_i(k) = [C_i^T(k) * \tilde{R}^{-1}(k)]/P_i(k/k)$$

$$P_i(k/k) = P_i(k-1/k-1) + C_i^T(k) * \tilde{R}(k)^{-1} * C_i(k)$$

(vi) Compute likelihood ratios

$$\alpha_i(k) = RES^T(k) * \tilde{R}^{-1}(k) * RES(k) + \alpha_i(k-1)$$

NOTE: Steps (iii) --> (vi) are performed in loop `w` number of times depending on which window has detected failure.

$C_i(0) = V_i(0) = P_i(0/0) = 0 \quad , \quad \alpha_i(0) = -12 * \ln(\text{Priori}_i)$

DECIDE: Find the minimum α and check failure level constraint $\hat{m}_i > 1.\sigma_i$

==> isolate failed sensor

RECONF (-1): Reconfigure system for any new `failed` sensor

Check if system can operate with remaining set --> else ABORT

GTOI : i) compute a/c latitude & longitude

(ii) compute rate-gyro compensation terms.

(iii) compute gravity vector

---End of Time `k`----

FINDS1 (Rotational Kinematics)

Number of states, NX = 3

State vector, $\hat{x} = [\hat{\phi}, \hat{\theta}, \hat{\psi}]^T$

Number of biases, NB = 3

Bias vector, $\hat{b} = [\hat{b}_p, \hat{b}_q, \hat{b}_r]^T$

Number of measurement types, NY = 3

Measurement vector, $y = [IMU_\phi, IMU_\theta, IMU_\psi]^T$

Number of inputs, NU1 = 3

Input vector, $u = [p, q, r]^T$

---New Time Iteration Start: time `k`---

<u>READFL</u>: Read the NFF input sensors $u_i^n(k)$, and the NFF measurement sensors, $y_j^n(k)$, $i=1,2,3, j = 1,2,3; n = 1,2$

<u>EKFN1(2)</u>: UPDG --→ Update input transition matrix $B(\hat{x}(k-1)$:

The differences from FINDSCOMP:

$B(\hat{x}(k-1)) = \Delta \cdot T_{ER}(\hat{x}(k-1))$

and the only other difference from FINDSCMP:

$A = I$

<u>EKFN1(1)</u>:

The differences from FINDSCMP are the following measurement partials:

$$H(\hat{x}(k-1) = \begin{bmatrix} 1/\sigma_\phi & 0 & 0 \\ 0 & 1/\sigma_\theta & 0 \\ 0 & 0 & 1/\sigma_\psi \end{bmatrix}$$

<u>FINDS2: (Translational Dynamics)</u>

Number of states, NX = 8

State vector, $\hat{x} = [\hat{r}_x, \hat{r}_y, \hat{r}_z, \hat{\dot{r}}_x, \hat{\dot{r}}_y, \hat{\dot{r}}_z, \hat{w}_x, \hat{w}_y]^T$

Number of biases, NB = 3

Bias vector, $b = [\hat{b}_{ax}, \hat{b}_{ay}, \hat{b}_{az}]^T$

Number of measurement types, NY = 4

Measurement vector, $y = [MLS_{az}, MLS_{e\ell}, MLS_{rn}, IAS]^T$

Number of inputs, NU1 = 3

Input vector, $u = [ax, ay, az]^T$

---Start of New Time Tick: (time `k`)---

<u>READFL</u>: Read the NFF input sensors $u_i^n(k)$, and the NFF measurement sensors, $y_J^n(k)$; i=1,3 ; j=1-4 ; n=1-2

<u>EKFN1(2)</u>: UPDB ---→ Update input transition matrix $B(\hat{x}(k-1))$:

The differences from FINDSCMP:

$$B(\hat{x}(k-1) = \begin{bmatrix} \frac{\Delta^2}{2} & T_{GB}(\hat{x}(k-1)) & \frac{\Delta^2}{2} I \\ \Delta & T_{GB}(\hat{x}(k-1) & \Delta I \\ 0 & & 0 \end{bmatrix}$$

where $\hat{\phi}(k-1)$, $\hat{\theta}(k-1)$ and $\hat{\psi}(k-1)$ in the evaluation of $T_{GB}(\hat{x}(k-1))$ are supplied by FINDS1.

$$A = \begin{bmatrix} I & \Delta I & 0 \\ 0 & I & 0 \\ 0 & 0 & A_W \end{bmatrix}$$

<u>EKFN1(1)</u>: The difference from FINDSCMP: The rows corresponding to IMU measurements are deleted.

4. SUBPROGRAM FLOW CHARTS

This section of the User's Guide contains signal flow and processing diagrams of the key subprograms of FINDS. The figures have been arranged in a nested sequence of increasing level of detail. Wherever possible, a figure is supported by those next in sequence.

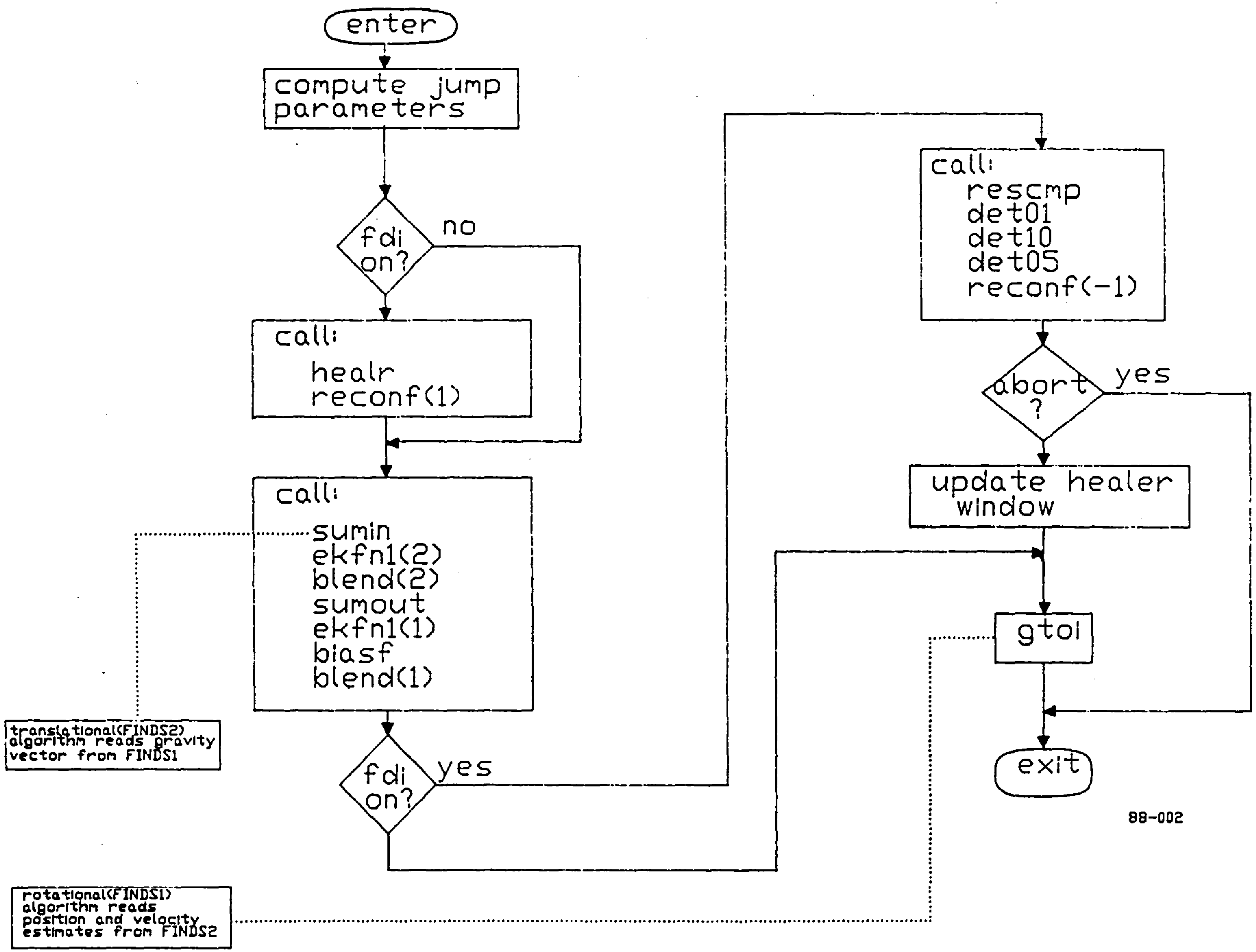

Figure 4.1: Flow Chart for Subprogram NAV

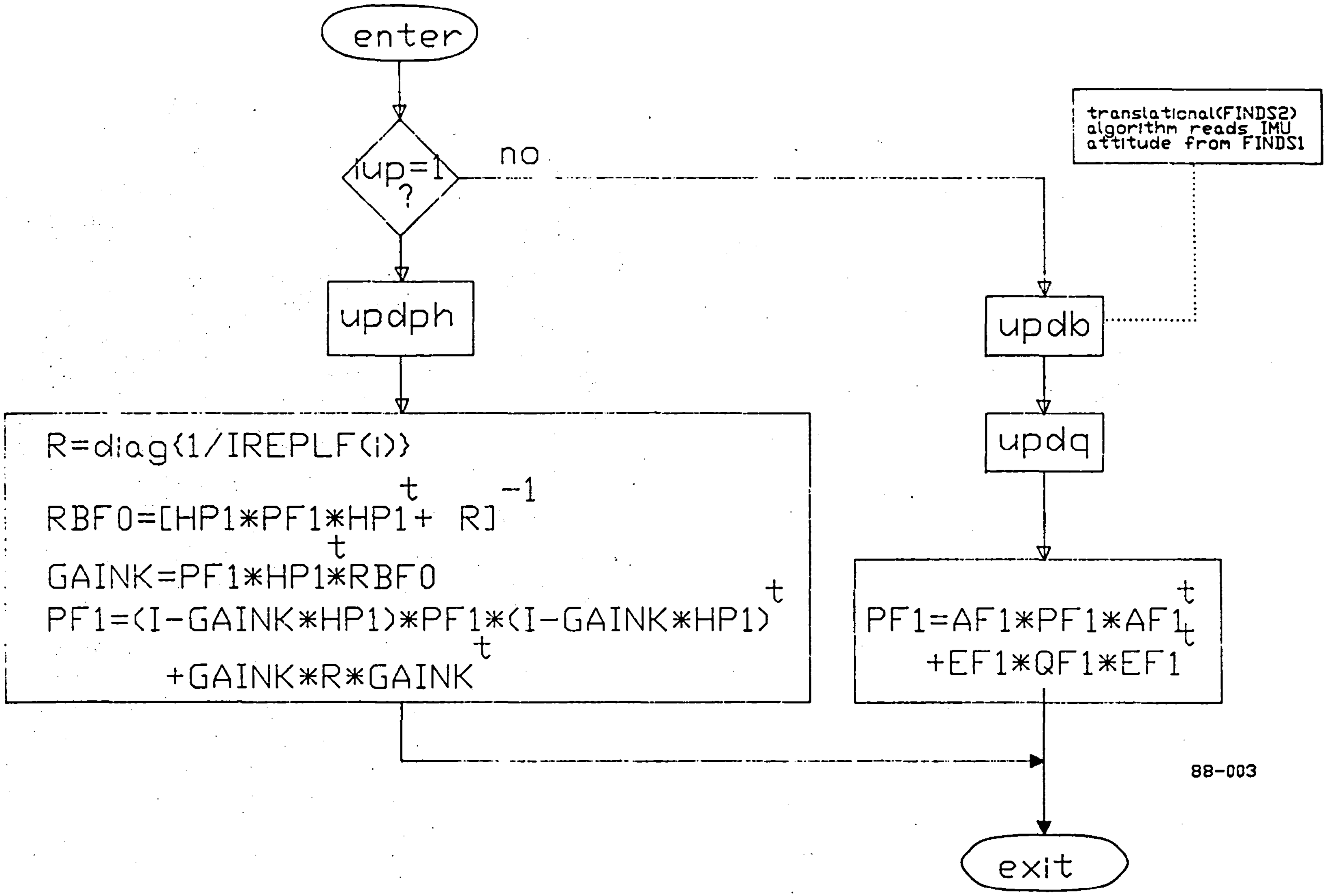

Figure 4.2: Flow Chart for Subprogram EKFN1

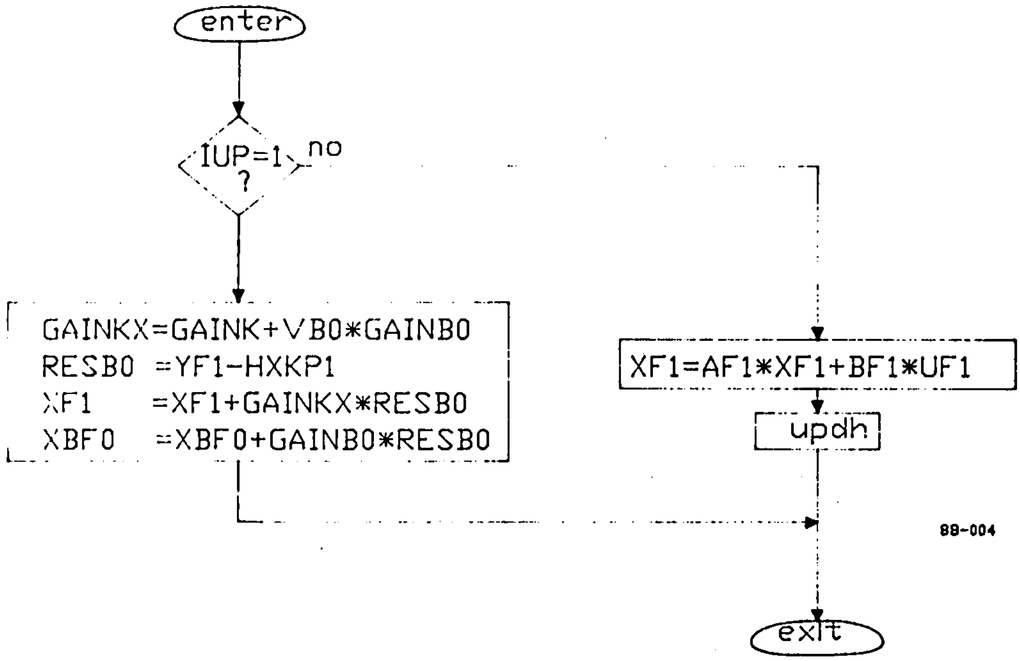

Figure 4.3: Flow Chart for Subprogram BLEND

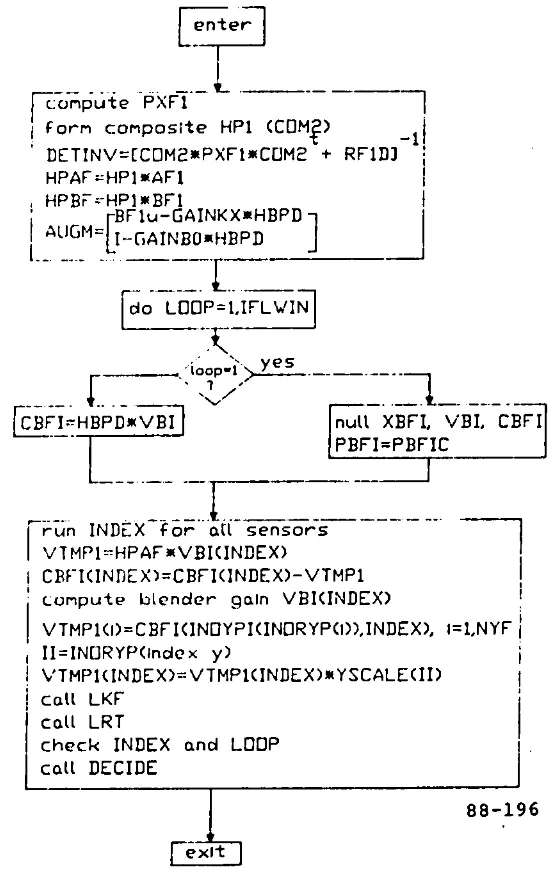

Figure 4.4: Flow Chart for Subprogram ISOLAT

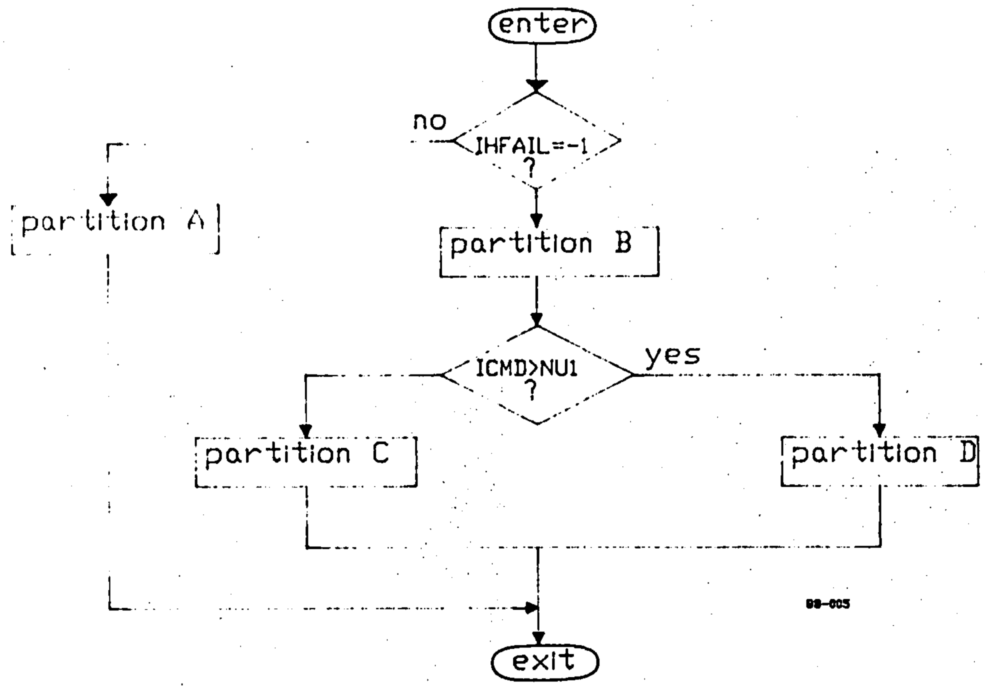

Figure 4.5: Flow Chart for Subprogram RECONF

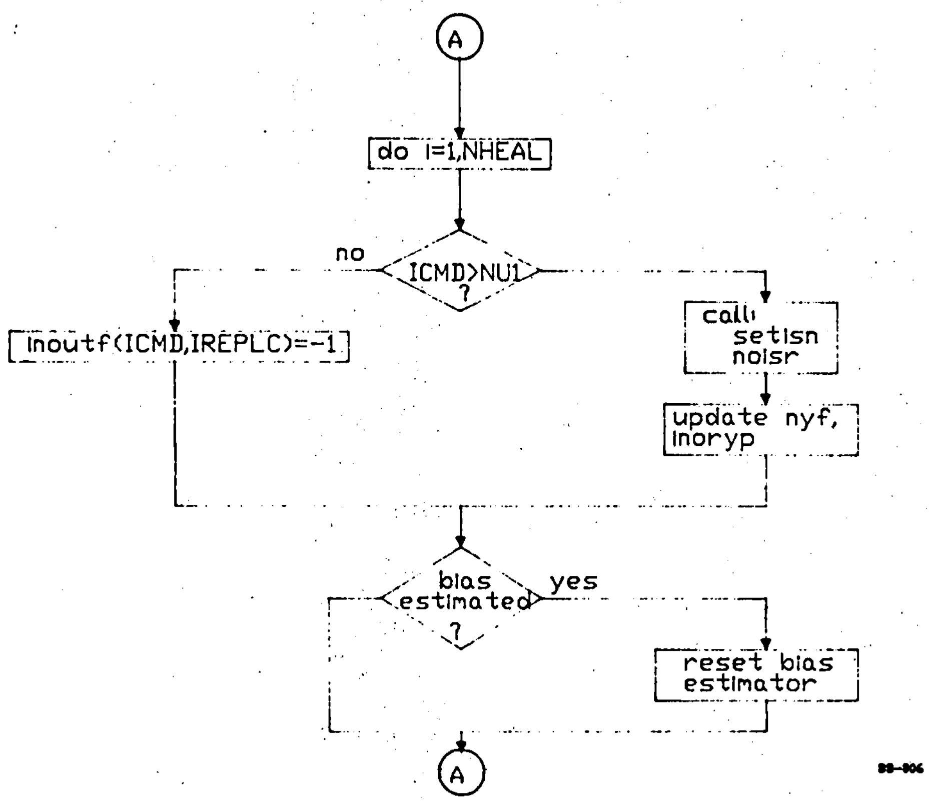

Figure 4.5a: Partition A of Subprogram RECONF

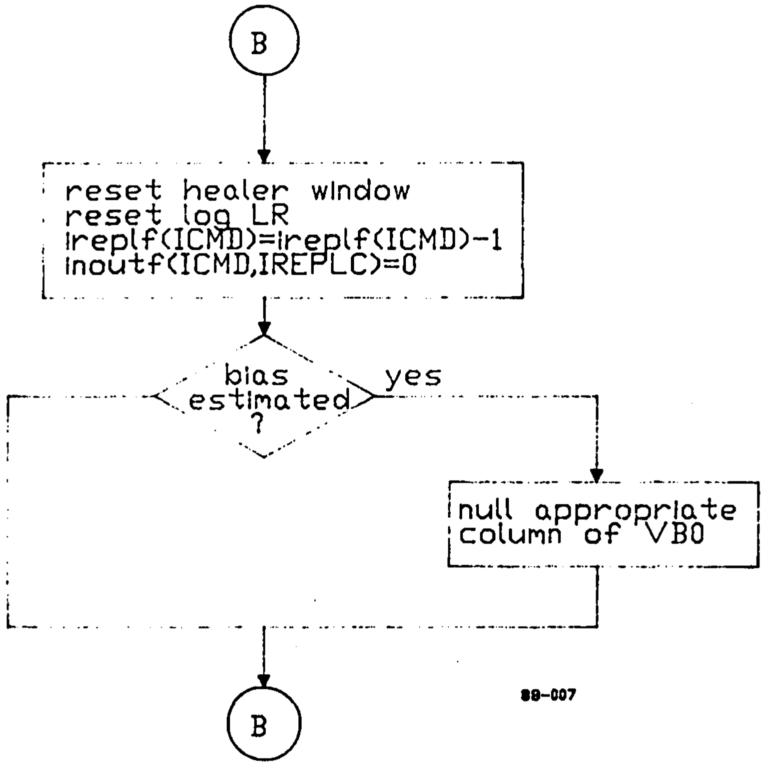

Figure 4.5b: Partition B of RECONF

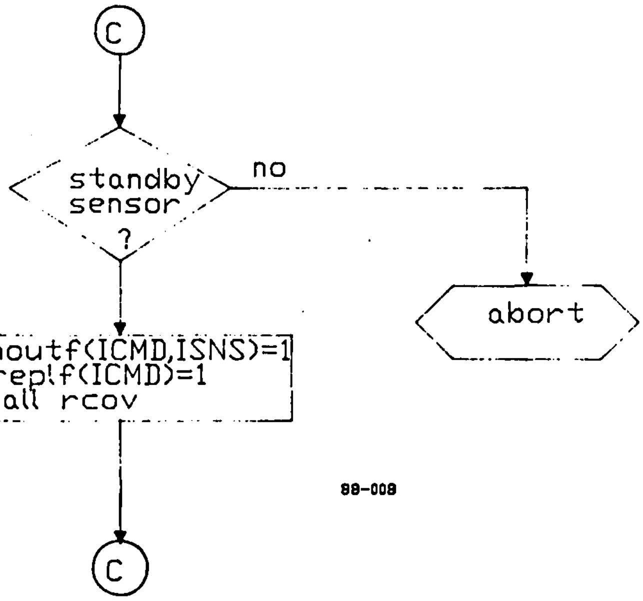

Figure 4.5c: Partition C of RECONF

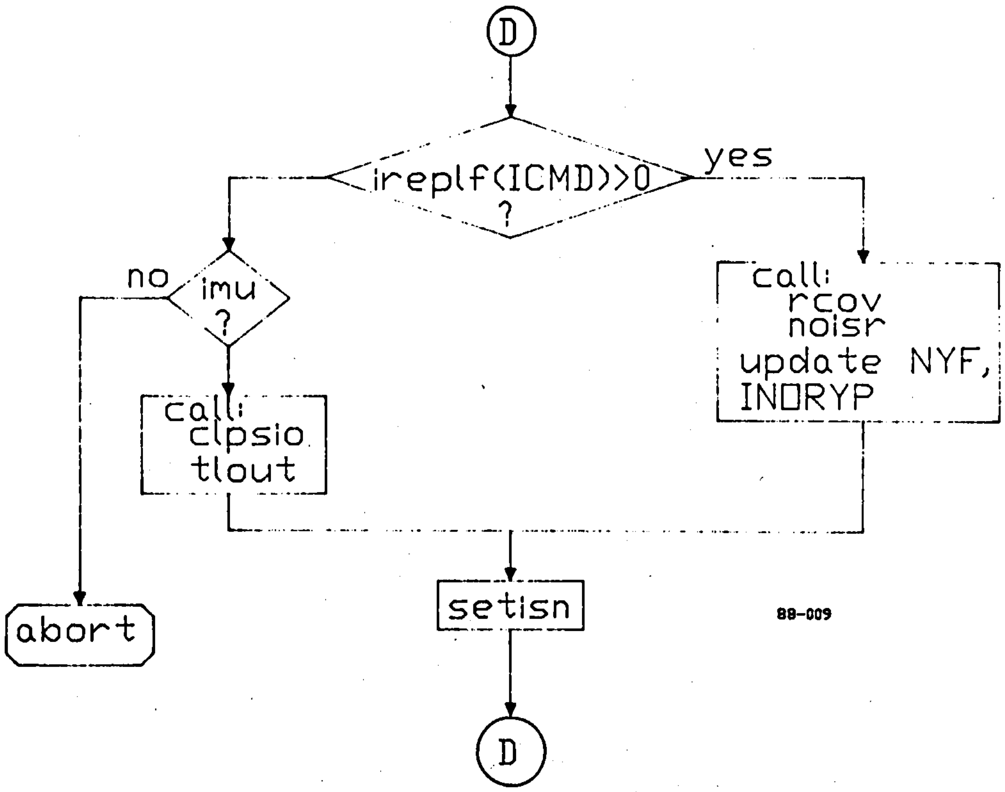

Figure 4.5d: Partition D of RECONF

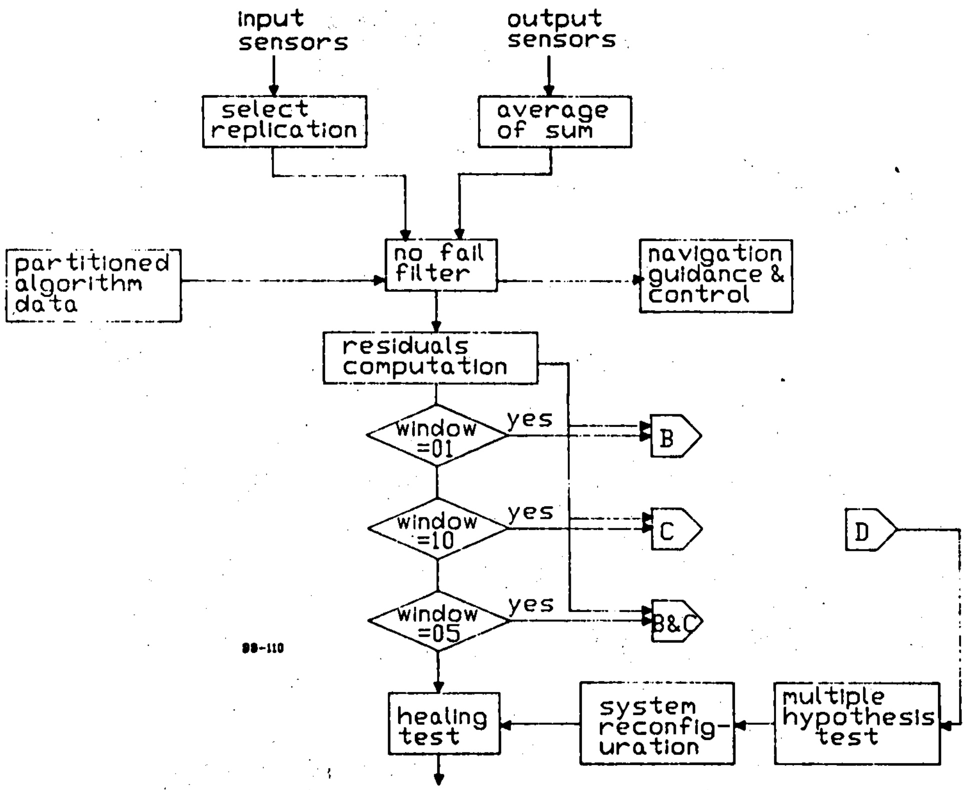

Figure 4.6: Hierarchical FDI Test

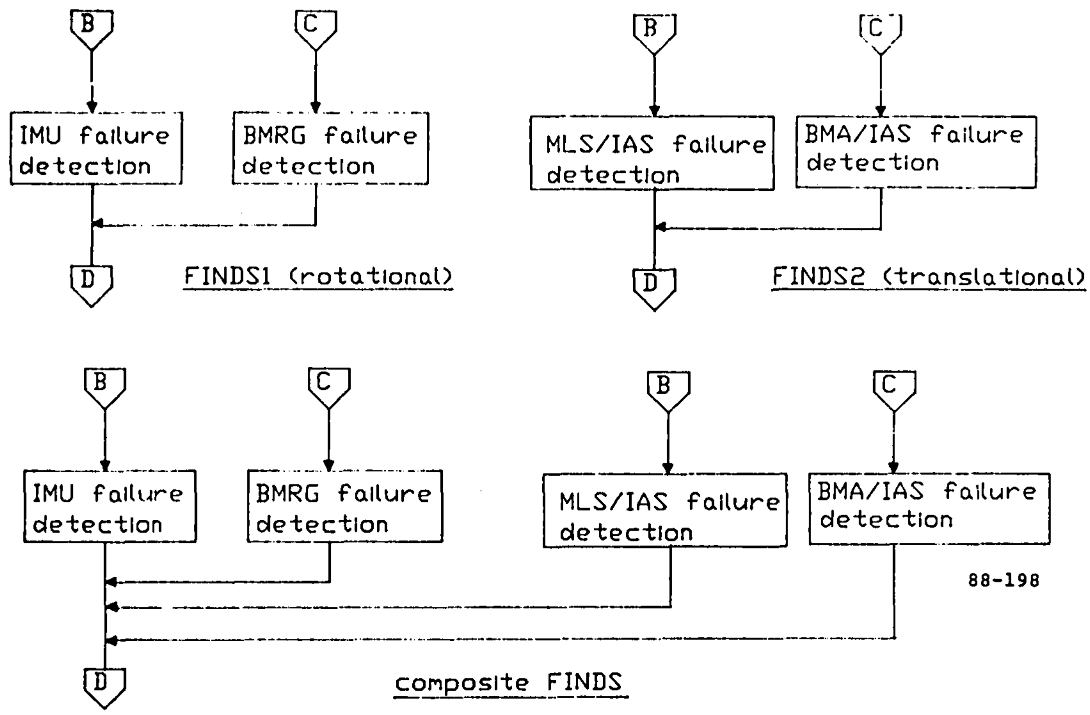

Figure 4.7: Isolation Logic

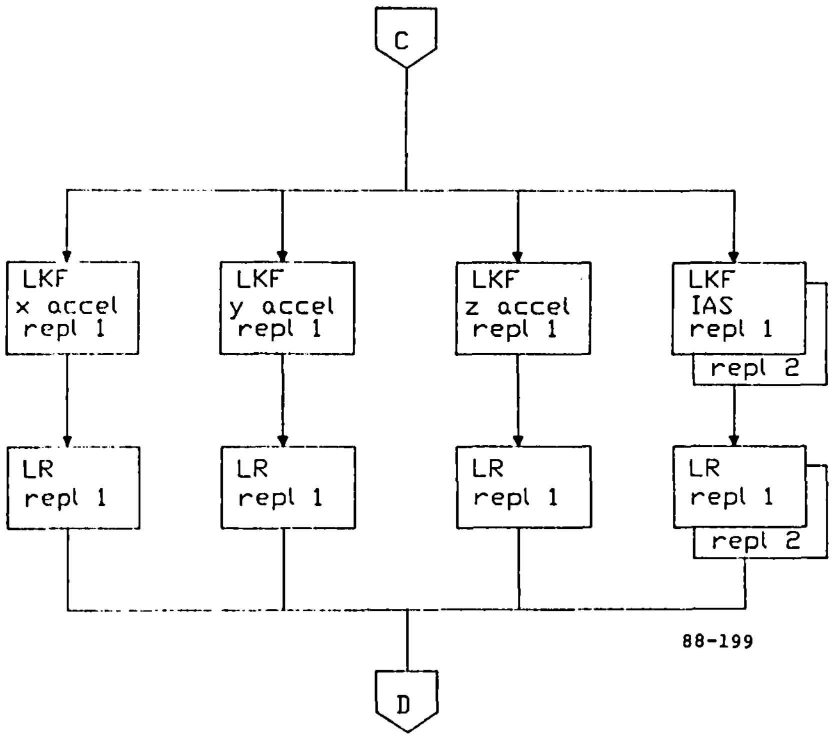

Figure 4.7a: BMA/IAS Failure Isolation

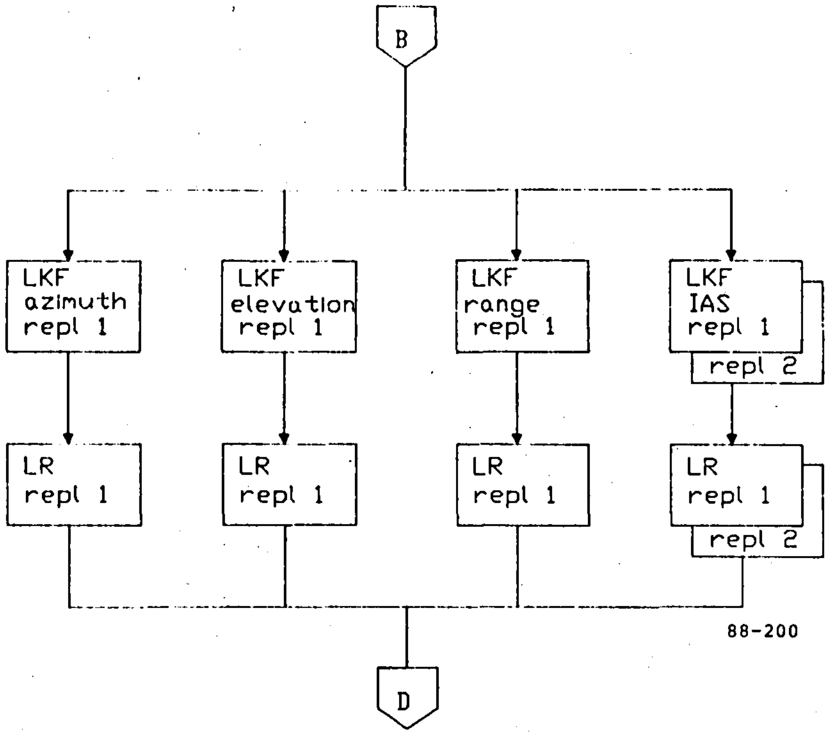

Figure 4.7b: MLS/IAS Failure Isolation

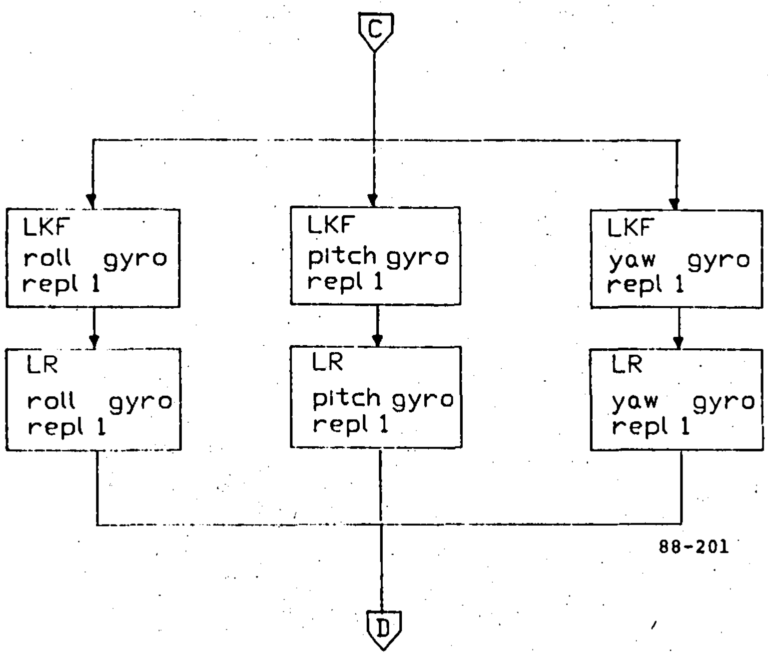

Figure 4.7c: RG Failure Isolation

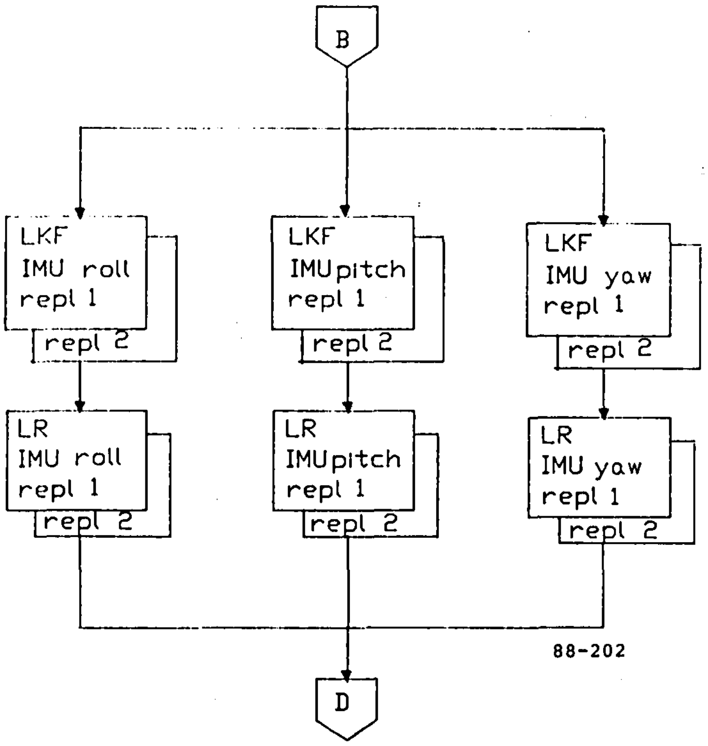

Figure 4.7d: IMU Failure Isolation

5. INPUT AND OUTPUT FILES

This section contains the descriptions of the input files required by and the output files generated by the FINDS program. In addition, the typical input design parameters are given in tables.

FINDS reads in the following files:

<u>ALGIN.DAT</u>

- o detector thresholds 01, 05, 10 windows
- o process noise SD
- o measurement noise SD 01, 05, 10 windows
- o wind model time contants

<u>RUNWAY.DAT</u>

- o initial aircraft latitude, longitude position
- o runway orientation relative to north
- o elevation and azimuth/range MLS locations
- o MLS and VOR antenna height above sea level

<u>FLDAT.NOF</u>

flight data time history of the NFF input (rate gyro, accelerometer) and measurement (MLS, IAS, IMU) sensors, two replications each for a total of 26 channels of data per record

Tables 5.1 and 5.2 depict typical values used as design parameters.

Table 5.1: Design Values for No-Fail Filter Noise Parameters

Variable	Noise S.D. Per Repl	Replications Used	Units
Process Noises			
Acc. Long.	0.05	1	m/s/s
Acc. Lat.	0.05	1	m/s/s
Acc. Vert.	0.05	1	m/s/s
Gyro Roll	0.05	1	deg/s
Gyro Pitch	0.05	1	deg/s
Gyro Yaw	0.05	1	deg/s
x-Wind-rw	0.10	N/A	m/s
y-Wind-rw	0.10	N/A	m/s
Measurement Noises			
MLS Azim.	0.06	1	deg
MLS Elev.	0.06	1	deg
MLS Range	6.00	1	m
IAS	3.00	2	m/s
INS Roll	0.25	2	deg
INS Pitch	0.50	2	deg
INS Yaw	0.30	2	deg

Table 5.2 Detector Design Values for Measurement Sensor Noise Parameters

Variable	Noise S.D. per Repl.	Replications Used	Units
MLS Azim	3.00E-02	1	deg
Elev	3.50E-02	1	deg
Range	5.50E-00	1	m
IAS	2.00E-00	2	m/s
INS-Roll	1.30E-01	2	deg
Pitch	1.50E-01	2	deg
Yaw	5.00E-01	2	deg

The following files are written by the program during execution:.

CHNGREP.DAT
sensor failure data (index, replication, time) for post processing

RUNNEW.PLT
time history of NFF states: position, velocity, attitude, and horizontal steady winds

RUNNEW.TLN
summary of events during the course of execution

LRT01.PLT, LRT05.PLT, LRT10.PLT
time history of likelihood ratio and measurement sensor residuals for detection windows 1, 5, and 10, respectively

EXPRES.PLT
expanded residual time history for those sensors with replications (IAS, IMU)

GTOI.XF1
time history of position and velocity states

SUMIN.UF1
time history of gravity vector

IMU.XF1
time history of attitude states

Note:

a) The partitioned algorithms FINDS1 and FINDS2 will read the same input as described above for FINDSCMP. In addition, GTOI.XF1 is input for FINDS1 and IMU.XF1 and SUMIN.UF1 are both input for FINDS2

b) Both FINDS1 and FINDS2 write a subset of the output shown above for FINDSCMP according to the table shown:

Algorithm	States	Residuals
FINDSCMP	position, velocity, attitude, wind, accelerometer bias, gyro bias	MLS, IAS, IMU
FINDS1	attitude, gyro bias	IMU
FINDS2	position, velocity, wind, accelerometer bias	MLS, IAS

6. PROGRAM VARIABLE INDEXING TABLES

This section describes the array indexing convention used in the FINDS software. These tables include the following array variables: NFF state and measurement vectors, process noise input vector, and the measurement vector.

Table 6.1: NFF Absolute State Indexing Convention

Program Arrays: XF1

	Array Index	State Variable	Program Units
FINDSCMP			
	1	x_{rw}	m
	2	y_{rw}	m
	3	z_{rw}	m
	4	$\dot{x}_{rw}$	m/s
	5	$\dot{y}_{rw}$	m/s
	6	$\dot{z}_{rw}$	m/s
	7	ϕ	radians
	8	θ	radians
	9	ψ	radians
	10	x_w	m/s
	11	y_w	m/s
FINDS1			
	1	ϕ	radians
	2	θ	radians
	3	ψ	radians
FINDS2			
	1	x_{rw}	m
	2	y_{rw}	m
	3	z_{rw}	m
	4	$\dot{x}_{rw}$	m/s
	5	$\dot{y}_{rw}$	m/s
	6	$\dot{z}_{rw}$	m/s
	7	x_w	m/s
	8	y_w	m/s

Table 6.2: NFF Absolute Measurement Indexing Convention

Program Arrays: RESB0, RF1D01, RF1D05, RF1D10, YF1, YSCALE, INOYP, INOYPI, SIG (latter part), SIGD01 (latter), SIGD05 (latter), SIGD10 (latter), HXKP1

	Array Index	Measurement Name	Program Units
FINDSCMP			
	1	MLS Azimuth	radians
	2	MLS Elevation	radians
	3	MLS Range	m
	4	IAS	m/s
	5	IMU Roll	radians
	6	IMU Pitch	radians
	7	IMU Yaw	radians
FINDS1			
	1	IMU Roll	radians
	2	IMU Pitch	radians
	3	IMU Yaw	radians
FINDS2			
	1	MLS Azimuth	radians
	2	MLS Elevation	radians
	3	MLS Range	m
	4	IAS	m/s

Table 6.3: NFF Absolute Input Indexing Convention

Program Arrays: UF1, INDUP, XBFO

	Array Index	Input Name	Program Units
FINDSCMP			
	1	a_x	m/s^2
	2	a_y	m/s^2
	3	a_z	m/s^2
	4	p	radians/s
	5	q	radians/s
	6	r	radians/s
FINDS1			
	1	p	radians/s
	2	q	radians/s
	3	r	radians/s
FINDS2			
	1	a_x	m/s^2
	2	a_y	m/s^2
	3	a_z	m/s^2

Table 6.4: NFF Process Noise Indexing Convention

Program Arrays: QF1, SIG (former part), SIGD01 (former), SIGD05, (former), SFGD10 (former)

	Array Index	Name	Program Units
FINDSCMP			
	1	a_x	m/s^2
	2	a_y	m/s^2
	3	a_z	m/s^2
	4	p	radians/s
	5	q	radians/s
	6	r	radians/s
	7	x_w	m/s
	8	y_w	m/s
FINDS1			
	1	p	radians/s
	2	q	radians/s
	3	r	radians/s
FINDS2			
	1	a_x	m/s^2
	2	a_y	m/s^2
	3	a_z	m/s^2
	4	x_w	m/s
	5	y_w	m/s

Program Arrays: INOBP, INOBPS, IYNAME, IYUNIT, CNVRF, PBFOI, PBFIC, IFAILT, BTHRSH, FTHRSH, DTMRSH, INDUTF, IREPLF

	Array Index	Sensor Type	Program Units
FINDSCMP			
	1	a_x	m/s^2
	2	a_y	m/s^2
	3	a_z	m/s^2
	4	p	radians/s
	5	q	radians/s
	6	r	radians/s
	7	MLS Azimuth	radians
	8	MLS Elevation	radians
	9	MLS Range	m
	10	IAS	m/s
	11	IMU ϕ	radians
	12	IMU θ	radians
	13	IMU ψ	radians
FINDS1			
	1	p	radians/s
	2	q	radians/s
	3	r	radians/s
	4	IMU ϕ	radians
	5	IMU θ	radians
	6	IMU ψ	radians
FINDS2			
	1	a_x	m/s^2
	2	a_y	m/s^2
	3	a_z	m/s^2
	4	MLS Azimuth	radians
	5	MLS Elevation	radians
	6	MLS Range	radians
	7	IAS	m/s

Table 6.6: Replicated Sensor Indexing Convention

Program Arrays: XBFI, PBFI, RESBI, CBFI, ICNTSN, PRIORI, ALAMDA

	Array Index	Sensor Type/Repl.	Program Units
FINDSCMP			
	1	a_x-n*	m/s^2
	2	a_y-n*	m/s^2
	3	a_z-n*	m/s^2
	4	p -n*	radians/s
	5	q -n*	radians/s
	6	r -n*	radians/s
	7	MLS Azim-n*	radians
	8	MLS Elev-n*	radians
	9	MLS Rng-n*	m
	10	IAS-1	m/s
	11	IMU ϕ-1	radians
	12	IMU θ-1	radians
	13	IMU ψ-1	radians
	14	IAS-2	m/s
	15	IMU ϕ-2	radians
	16	IMU θ-2	radians
	17	IMU ψ-2	radians
FINDS1			
	1	p-n*	radians/s
	2	q-n*	radians/s
	3	r-n*	radians
	4	IMU ϕ-1	radians
	5	IMU θ-1	radians
	6	IMU ψ-1	radians
	7	IMU ϕ-2	radians
	8	IMU θ-2	radians
	9	IMU ψ-2	radians
FINDS2			
	1	a_x-n	m/s^2
	2	a_y-n	m/s^2
	3	a_z-n	m/s^2
	4	MLS Azim-n	radians
	5	MLS Elev-n	radians
	6	MLS Rng-n	m
	7	IAS-1	m/s
	8	IAS-2	m/s

* 'n' refers to the replication currently in use by the NFF (i.e., 1 or 2)

Table 6.7: Replicated Measurement Indexing Convention

Program Arrays: INORYP

	Array Index	Meas. Sensor Type/Repl.	Program Units
FINDSCMP			
	1	MLS Azim-n*	radians
	2	MLS Elev-n*	radians
	3	MLS Rng-n*	m
	4	IAS-1	m/s
	5	IMU ϕ-1	radians
	6	IMU θ-1	radians
	7	IMU ψ-1	radians
	8	IAS-2	m/s
	9	IMU ϕ-2	radians
	10	IMU θ-2	radians
	11	IMU ψ-2	radians
FINDS1			
	1	IMU ϕ-1	radians
	2	IMU θ-1	radians
	3	IMU ψ-1	radians
	4	IMU ϕ-2	radians
	5	IMU θ-2	radians
	6	IMU ψ-2	radians
FINDS2			
	1	MLS Azim-n*	radians
	2	MLS Elev-n*	radians
	3	MLS Rng-n*	m
	4	IAS-1	m/s
	5	IAS-2	m/s

*'n' refers to the replication currently in use by the NFF (i.e., 1 or 2)

7. SUBPROGRAM DESCRIPTION AND TABLES

This section contains a description of all subprograms in FINDS. Table 7.1 is a "quick" reference list of each subprogram and its associated "calls to" and "called by" programs. Subsequent paragraphs explain the specific function of each subprogram and list its associated common blocks.

TABLE 7.1

SUBPROGRAMS

Called by:	Name	Calls to:
Main program (FINDS/FINDS1/FINDS2)	READFL	
Main program	NAV	HEALR, RECONF, SUMIN, EKFNI, BLEND, SUMOUT, BIASF, RESCMP, DET01, DET05, DET10, GTOI
Main program	INITG	BUBBL2, INITXF, UPDB, VEQUAL, GTOI
INITG	INITXF	
NAV	SUMIN	
NAV	SUMOUT	
NAV, INITG	GTOI	
NAV	EKFN1	UPDPH, PDMINV, MATIA, MAT3 VSCALE, MATS, MADD, UPDB, UPDQ, PD3NV1, PMAXB, PMABAT, PMABT2, PMAPB, PD4NV1,
NAV	BIASF	VSUB, MEQUAL, MAT1A, MATVAC, VSCALE, MSUB, MATXYT, MADD, PDMINV, PMBEA, PMAXB, PMAXV, YSCALE, PMAMB, PMABT, PMAPB, PD3NV1, PD4NV1
NAV	BLEND	MAT1A, MADD, MATVC2, UPDH, PMAXB, PMAPB, PMAXV2
NAV	DET01	MEQUAL, MAT3B, ISOLAT, PMBEA, PMVTAV
	DET05	MEQUAL, MAT3B, ISOLAT, PMBEA, PMVTAV
	DET10	MEQUAL, MAT3B, ISOLAT, PMBEA, PMVTAV
RECONF	SETISN	
INITG, EKFNI	UPDB	
EKFNI	UPDQ	
BLEND	UPDH	
CLIPSIO, EKFNI	UPDPH	
NAV	RESCMP	
DET01, DET05, DET10	ISOLAT	MAT1A, PDMINV, VEQUAL, VSUB, VADD, LKF, DECIDE, MEQUAL, TRANS2, MATXYT, MADD, MADZ, MSUB, MATLN2, MATVEC, LRT

Called by:	Name	Calls to:
ISOLAT	LKF	
ISOLAT	LRT	MAT3B
ISOLAT	DECIDE	VEQUAL, TLOUT, VMPRT, VMPRT2, MINIM2, MINIM3
NAV	RECONF	PNTINV, RCOV, SET1SN, CLPS10, TLOUT, IMEG2, NOISR, MATLN2,
RECONF	CLPSIO	RCOV, PMTINV, IMTCG2, CLPSBE, NOISR, UPDPH
RECONF, CLPSIO	NOISR	
CLPSIO	CLPSBE	ADJTPB, MATCG2, PNT1NV, 1MTCG2, MATCG3
CLPSBE	ADJTBP	PNTINV, IMTCG2
CLPSIO, RECONF	RCOV	VMPRT, MATLN2, VMPRT2, MATLN3
NAV	HEALR	BUBBL2, TLOUT, LRTHLR
HEALR	LRTHLR	
Main Program, DECIDE, RECONF, HEALR		

Includes files `FINDSCMP.FOR` , `FINDS1.FOR` , `FINDS2.FOR`.

NOTE: (a) The exact-dimensioned versions of FINDS1 & FINDS2 are summarized here. The documentation for the `NDIM' dimensioned versions is along the same lines as for FINDSCMP.

(b) Everything is common to all 3 files except where specified by file-name.

(c) Notation in this document is as follows:
func --→ function (of routine)
refs --→ refers (other routines it refers to)
refby --→ referred by (other routines it gets called by)
comm --→ common blocks (used in the routine)
args --→ variables in the argument list

I. DESCRIPTION OF SUBROUTINES

name: FINDS/FINDS1/FINDS - (Main Program)
func: Coordinates the run-time operation of the FINDS algorithm. FINDS1 is the rotational kinematics portion and FINDS2 is the translational dynamics portion of the composite algorithm. Initializes program variables, reads-in first iteration of flight data and initializes the filter. The basic run-time loop consists of reading in one iteration of flight data (READFL) and passing control to NAV which coordinates the FTN/FDI algorithm.
refs: INITG, READFL, TLOUT, NAV
comm: FINDSCMP --→ EARTH, MCONCO, SYNC, IMLS, MLSALL, PSIR, CNTROL, ABRTCM
FINDS1 --→ EARTH, MCONCO, SYNC, IMLS, PSIR, CNTROL, ABRTCM
FINDS2 --→ SYNC, MLSALL, CNTROL, ABRTCM

name: READFL
func: Flight data interface routine -- reads in the flight data from binary data file, assigns data to the various sensor variables and converts data to program working units (i.e, radians, m, m/s, m/s^2). Also checks for data dropouts and "fixes" them by substituting data from previous iteration.
call: Call READFL
args: None
refs: None
refby: FINDS/FINDS1/FINDS2
comm: FINDSCMP --→ SYNC, MCONCO, RGOUT, LAOUT, AGOUT, ASOUT, MLOUT, NAMES, RDLOCL
FINDS1 --→ SYNC, MCONCO, RGOUT, AGOUT, NAMES, RDLOCL
FINDS2--→ SYNC, MCONCO, LAOUT, ASOUT, MLOUT, FLTIN, NAMES, RDLOCL

name: NAV
func: Executive program which coordinates the no-fail filter (NFF) (or fault tolerant navigator FTN) and failure detection (isolation (FDI) modules, (see attached flow chart)
call: Call NAV
args: None

refs: HEALR, RECONF, SUMIN, EKFN1, BLEND, SUMOUT, BIASF, RESCMP, DET01, DET10, DET05, GTOI (note: FINDS2 does not contain routine GTOI)
refby: FINDS/FINDS1/FINDS2
comm: SYNC, CNTROL, ABRTCM, EKF1, HEALCM, SYSXB0, JUMPCM, DTSYNC, HFCOM

name: INITG
func: Sets program flags and initializes parameters used in the NFF, FDI and reconfiguration modules. The initialization process is in two passes; the first pass configures the system dimensions based on sensor replications used and also sets the healer parameters. The second pass sets the initial conditions for the NFF states and initializes the NFF measurement and covariances.
call: Call INITG
args: None
refs: FINDSCMP --→ BUBBL2, VEQUAL, INITXF, UPDB, GTOI
FINDS1 --→ BUBBL2, INITXF, UPDB, GTOI
FINDS2 --→ BUBBL2, INITXF, UPDB
refby: FINDS/FINDS1/FINDS2
comm: SYSX1, SYSYW1, SYSU1, EKF1, EKBF0, SYSXB0, CMPSTF, DETXBI, SYNC, MCONCO, FILTRT, INITVL, DETINF, CNTROL, FILTIC, YOBSRV, MAIN1, HEALCM In addition, FINDSCMP/FINDS2 contain blocks SIGTAU, ASOUT and FINDS1 contains block SIG)

name: INITXF
func: Uses the first iteration of the flight data to compute the NFF state initial conditions. A/C position is calculated using a reconstruction algorithm from the MLS emasurements. Velocity is estimated by resolving the averaged IAS measurement in the appropriate axis. A/C attitude initial estimates are obtained by averaging the replicated IMU measurements. Initial horizontal winds are estimated to be zero.
call: Call INITXF
args: None
refs: None
refby: INITG
comm: FINDSCMP --→ FILTRT, MCONCO, EKF1, ASOUT, MLSALL, AGOUT, MLOUT, PSIR
FINDS1 --→ FILTRT, AGOUT PSIR, EKF1
FINDS2 --→ FLTIN, MLOUT, ASOUT, MLSALL, MCONCO, FILTRT, EKF1

name: SUMIN
func: Provides a proper set of inputs to the NFF. The input vector is formed as follows:
1) Only on replication of all input sensors is in active mode; the second replication is kept either in standby or in failed status.
2) the input vector, UF1, is formed such that trapezoidal integration is performed, i.e., U(k) = 0.5 * {u(k) + u(k-1)]
3) current estimates of input sensor biases (XBF0) are subtracted from UF1.
4) FINDSCMP, FINDS1 --→ rate gyro measurements are compensated for earth and platform rates
5) FINDSCMP, FINDS2 --→ the gravity vector (Gx, Gy, Gz) expressed in the G-frame is added to the end of UF1.
6) FINDSCMP --→ UF1 ≡ $[Ax, Ay, Az, P, Q, R, Gx, Gy, Gz]^T$

FINDS1 --→ $UF1 \equiv [P, Q, R]^T$
FINDS2 --→ $UF1 \equiv [Ax, Ay, Az, Gx, Gy, Gz]^T$

7) In the split versions, FINDS1 generates the gravity vector in GTOI which is then transferred over to FINDS2 and used there.

call: Call SUMIN
args: None
refs: None
refby: NAV
comm: FINDSCMP --→ MAIN1, RGOUT, LAOUT, EKBFO, SYSU1, SYSXBO, FILTRT, SYNC, EARTH, PSIR, TRBER, LATLON, SUMLOC

name: SUMOUT
func: Forms a set of measurements (YF1) to be used by the NFF

1) each sensor replication has an active or failed or standby status, and the number of available active replicated measurements are averaged
2) each measurement is normalized by the expected variance of that signal (scale factor is set in INITG)
3) psi measurements are compensated for runway yaw in FINDSCMP and FINDS1
4) FINDSCMP --→ $YF1 \equiv [Azim, Elev, Rng, IAS, Phi, Theta, Psi]^T$
   FINDS1 --→ $YF1 \equiv [Phi, Theta, Psi]^T$
   FINDS2 --→ $YF2 \equiv [Azim, Elev, Rng, IAS]^T$

call: Call SUMOUT
args: None
refs: None
refby: NAV
comm: FINDSCMP --→ PSIR, ASOUT, AGOUT, MLOUT, SYSYW1, FILTRT, YOBSRV, DETXBI
FINDS1 --→ PSIR, AGOUT, SYSYW1, FILTRT, YOBSRV, DETXBI
FINDS2 --→ ASOUT, MLOUT, SYSYW1, FILTRT, YOBSRV, DETXBI

name: GTOI (not in FINDS2)
func: Forms estimates for inertial position, velocity and acceleration, and runway acceleration. Also computes the a/c's current longitude and latitude along with their rates of change. In addition, coriolis and centripetal correction terms for compensating the platform gravity force are also computed. [NOTE: in FINDS1, this routine needs the a/c position and velocity estimates generated by FINDS2]
call: Call GTOI
args: None
refs: None
refby: NAV, INITG
comm: MAIN1, FILTRT, RGOUT, SYSU1, EKF1, TRBER, MCONCO, EARTH, IMLS, PSIR, LATLON, PQRDEG, GRVYTC, GTOILC

name: EKFN1
func: Represents the bias-free filter portion of the NFF and is implemented as an extended Kalman filter (EKF). Covariance propagation of the stabilized normal equations is performed. The state estimates, XF1, are not computed in this routine. (see attached flow chart)
call: Call EKFN1 (Iup)

args: Iup -- integer in ; update/propagate flag (1 ==> update, 2 ==> propagate)
refs: FINDSCMP --→ UPDPH, PDMINV, MAT1A, MAT3, VSCALE, MAT2, MADD, UPDB, UPDQ
FINDS1 --→ PD3NV1, PMAXB, PMABAT, VSCALE, PMABT2, PMAPB, UPDB, UPDQ
FINDS2 --→ UPDPH, PD4NV1, PMAXB, PMABAT, VSCALE, PMABT2, PMAPB, UPDB, UPDQ
refby: NAV
comm: FINDSCMP --→ MAIN2, SYSX1, SYSYW1, SYSU1, EKF1, SYSXB0, SYSYB0, FILTRT, TSTORE, CNTROL, EKFBIA, JUMPCM
FINDS1 --→ MAIN2, SYSX1, SYSYW1, SYSU1, EKF1, SYSXB0, SYSYB0, FILTRT, CNTROL, EKFBIA, JUMPCM
FINDS2 --→ SYSX1, SYSY1, SYSU1, EKF1, SYSXB0, FILTRT, CNTROL, EKFBIA, JUMPCM, EKFBLN, EKFWRK

name: <u>BIASF</u>
func: Implements the bias filter portion of the NFF. There are no bias filter dynamics; hence no propagation step is required and this routine is called only during the update mode of the NFF.
call: Call BIASF
args: None
refs: FINDSCMP --→ VSUB, MEQUAL, MAT1A, MATVEC, VSCALE, MSUB, MATXYT, MADD, PDMINV
FINDS1 --→ VSUB, PMBEA, PMAXB, PMAXV, YSCALE, PMAMB, PMABT, PMAPB, PD3NV1
FINDS2 --→ VSUB, PMBEA, PMAXB, PMAXV, VSCALE, PMAMB, PMABT, PMAPB, PD4NV1
refby: NAV
comm: MAIN1, SYSX1, SYSYW1, SYSU1, EKBF0, SYSXB0, GBLEND, YOBSRV, FILTRT, EKF1, EKFBIA, LRTINV, DETCOV, JUMPCM, CNTROL
In addition to the above,
FINDSCMP --→ MAIN2, SYSYB0, TSTORE
FINDS1 --→ MAIN2, SYSYB0, BSFWRK
FINDS2 --→ BSFWRK

name: <u>BLEND</u>
func: Computes the bias and bias-free state estimates and "blends" them together to form the total state and bias estimates. Also forms the Kalman gain matrix. (see flow chart)
call: Call BLEND (Iup)
args: Iup -- integer in ; update/progagate flag (1 ==> update, 2 ==> propagate)
refs: FINDSCMP --→ MAT1A, MADD, MATVC2, UPDH
FINDS1, FINDS2 --→ PMAXB, PMAPB, PMAXV2, UPDH
refby: NAV
comm: SYSX1, SYSYW1, SYSU1, EKF1, EKBF0, SYSXB0, GBLEND, CMPSTF, DETINF, FILTRT, JUMPCM
In addition to the above,
FINDSCMP --→ MAIN2, TSTORE
FINDS1 --→ MAIN2
FINDS2 --→ EKFBLN, BLNDWK

name: DET01
func: Implements the failure detector of moving residual window 1 sample, i.e., the current filter residual. Peforms a Chi-square test on the NFF averaged measurement residual RESB0 and checks against set thresholds to detect failures. Calls isolation routine ISOLAT if failure is detected.
call: Call DET01
args: None
refs: FINDSCMP --→ MEQUAL, MAT3B, ISOLAT
FINDS1, FINDS2 --→ PMBEA, PMVTAV, ISOLAT
refby: NAV
comm: SYNC, SYSYW1, SYSU1, FILTRT, EKBF0, LRTINV, JUMPCM, LRTMAX, DTCT01, CNTROL, DETPRI

name: DET05
func: Implements the failure detector of moving residual window length 5 samples. Performs a Chi-square test on the moving average of RESB0 over the last 5 samples (incl. current residual).
call: Call DET05
args: None
refs: FINDSCMP --→ MEQUAL, MAT3B, ISOLAT
FINDS1, FINDS2 --→ PMBEA, PMVTAV, ISOLAT
refby: NAV
comm: SYNC, SYSYW1, SYSU1, FILTRT, EKBF0, LRTINV, JUMPCM, LRTMAX, DTCT05, CNTROL, DETPRI

name: DET10
func: Implements the failure detection of moving residual window length 10 samples. Performs a Chi-square test on the moving average of RESB0 over the last 10 samples (incl. current residual).
call: Call DET10
args: None
refs: FINDSCMP --→ MEQUAL, MAT3B, ISOLAT
FINDS1, FINDS2 --→ PMBEA, PMVTAV, ISOLAT
refby: NAV
comm: SYNC, SYSYW1, SYSU1, FILTRT, EKBF0, LRTINV, JUMPCM, LRTMAX, DTCT10, CNTROL, DETPRI

name: SETISN
func: Maintains the value of vector ICNTSN in which the ordering of elements corresponds to the absolute replicated sensor ordering (Table 6.6). The value of each element is the location in UF1 for the input elements (six for FINDSCMP, 3 for FINDS1/FINDS2), and the location in the expanded innovations for the rest of ICNTSN. ICNTSN provides a mapping between an absolute indexing scheme and a collapsed indexing scheme in the event of failures.
call: Call SETISN
args: None
refs: None
refby: RECONF
comm: DETINF, FILTRT, SYSU1, DETXBI

name: UPDB
func: Updates the discrete input weighting matrix BF1 and also evaluates and saves:
1) sines and cosines of the estimated Euler angles (in FINDS2, these are the estimates transferred over from FINDS1 at each iteration).
2) the transformation from the B to the R frame
3) the transformation from the R to the E frame (not in FINDS2).
call: Call UPDB
args: None
refs: None
refby: INITG, EKFN1
comm: MAIN1, TRBER, EULER, SYNC, SYSU1, EKF1, SYSX1

name: UPDQ
func: Updates the discrete process noise covariance matrix EF1. Assumes that UPDB has been called before this routine, hence transformation matrices Trb and Ter are current. In addition, for FINDSCMP and FINDS1, terms to represent the rate gyro errors due to scale factor and misalignment are added to the measurement noise variance.
call: Call UPDQ
args: None
refs: None
refby: EKFN1
comm: MAIN1, TRBER, SYNC, MCONCO, SYSX1, SYSYW1, UPDQLC
In addition to the above,
FINDSCMP --→ SIGTAU, PQRDEG
FINDS1 --→ SIG, PQRDEG
FINDS2 --→ SIGTAU

name: UPDH
func: Updates the nonlinear observations function H, called HXKP1
call: Call UPDH
args: None
refs: None
refby: BLEND
comm: YOBSRV, SYSX1, SYSYW1, EKF1, SYSU1, EKBFO, SYSXBO
In addition,
FINDSCMP, FINDS2 --→ MLSALL

name: UPDPH (not in FINDS1)
func: Updates the partial of H (i.e., HXKP1) w.r.t. XF1, called HP1. Not used in FINDS1 as HP1 is an identity matrix in that algorithm.
call: Call UPDPH
args: None
refs: None
refby: EKFN1, CLPSIO
comm: MAIN1, MLSALL, YOBSRV, SYSXBO, SYSU1, SYSYW1, CMPSTF, SYSX1, EKF1

name: RESCMP
func: Computes the expanded residuals sequence (RESBOC) from the residual sequence (RESBO) generated by the NFF. This sequence is the same as

the one which would have been generated had the filter been driven by all replications of the measurement sensors rather than their average value. This expanded residuals sequence is used in the failure isolation strategy.

call: Call RESCMP

args: None

refs: None

refby: NAV

comm: EKF1, YOBSRV, SYSYW1, DETINF, FILTRT, SYSU1, DTSYNC
In addition,
FINDSCMP ---→ ASOUT, AGOUT, MLOUT, PSIR
FINDS1 ---→ AGOUT, PSIR
FINDS2 ---→ ASOUT, MLOUT

name: ISOLAT

func: Implements a bank of first order filters and likelihood ratio computers in the isolation strategy. Each filter hypothesizes the occurrence of a failure at the beginning of the residual window (based on the length of the detector sequence which flagged the failure), and estimates the level of a bias jump failure by observing the expanded (and saved) residuals sequence over that window. The hypothesized failure is assumed to affect the NFF input measurements or output measurements only. Thus, a single failure cannot directly enter into BOTH an input and an ouput measurment.

A select subset of all first order filters is activated depending on which detector caught the failure. If the detector of window length 1 sample (DET01) signals the failure, then only the output sensor filters are activated. Similarly, if DET10 flags the failure, then only the input sensors (and the IAS sensor in the case of FINDSCMP/FINDS2) filters are activated. For DET05, no such assumptions are made and all of the sensors are equally "suspect."

The first order filters generate a sequence of failure compensated residuals which are used by the bank of likelihood ratio computers to compute the log likelihood of a singleton sensor failure (or a dual simultaneous failure in MLS sensors).

Subroutine ISOLAT functions as an executive of this bank of filter/LR computers. In the current version of FINDSCMP/FINDS2, only one replication of the MLS sensors is kept active and the other is in standby status (like the input sensors); hence, dual simultaneous MLS sensor failures are not considered in this routine or in DECIDE. (see attached flowchart)

call: Call ISOLAT (Iflwin)

args: Iflwin -- integer in ; length of detector window which flagged the failure (has value of either 1 or 5 or 10)

refs: MAT1A, PDMINV, VEQUAL, VSUB, VADD, LKF, DECIDE
In addition,
FINDSCMP ---→ MEQUAL, TRANS2, MATXYT, MADD, MATZ, MSUB, MATNL2, MATVEC, LRT

refby: DET01, DET05, DET10

comm: MAIN1, SYSX1, SYSYW1, SYSU1, SYSXB0, YOBSRV, EKF1, EDBF0, CMPSTF, DETXBI, DETINF, DCIDEI, DETYBI, INITVL, FILTRT, DTSYNC, DETCOV, DETLC3

In addition,
FINDSCMP --→ MAIN2, TSTORE, MULTDT, DETLC2
FINDS1 --→ MAIN2, DETWRK
FINDS2 --→ MULTDT, DETWRK, DETLC2

name: LKF
func: Provides the failure estimator structure in the isolation strategy. Implements a linear Kalman filter using the information form, and assumes a scalar state equation. The plant, measurements and filter equations are commented in the actual code in each algorithm. Generates a set of failure compensated residuals and also a "best" estimate of failure level for each suspect sensor.
call: Call LKF (Index, Ci, Istart)
args: Index -- Integer in ; points to particular sensor in question (has value based on Table 6.6 indexing)
Ci -- real in ; effective observations matrix (computed in ISOLAT)
Istart -- integer ; location in saved, expanded residual sequence RESBOC (has value between 1 and 10 depending on current location and Iflwin)
refs: None
refby: ISOLAT
comm: MAIN1, DETINF, DETXBI, DETYBI, DETLC3

name: LRT (only in FINDSCMP)
func: Computes the log likelihood ratios in the isolation strategy. The computations are as follows:
1) if loop = 1, A = -PHj. This initializes the log likelihood ratio A to -ln(PHj) at the start of the detection/decision residual window.
2) SUMI = RES^T * RTinv * RES
3) A = 0.5 * SUMI + A)
args: Loop -- integer in ; detection/decision window step (has values from 1 to Iflwin)
PHj --→ real in ; log of a-priori probability that the j'th sensor will fail
RES -- real in ; failure corrected innovations sequence from the j'th LKF
A -- real in out ; computed value of log likelihood ratio for j'th failure hypothesis.
refs: MAT3B
refby: ISOLAT
comm: MAIN1, DETINF, DETLC3

name: DECIDE
func: Chooses the most likely failure hypothesis by finding the smallest log likelihood ratio of those computed in LRT. For a chosen hypothesis, it checks for a minimum accepTable 6.6ailure level, else chooses the next likely hypothesis. Also, prints out various user messages.
call: Call DECIDE (Iflwin)
args: Iflwin -- integer in ; length of detector window which flagged the failure (either 1 or 5 or 10)
refs: FINDSCMP --→ VEQUAL, MINIM2, TLOUT, VMPRT

FINDS1 --→ VEQUAL, MINIM3, TLOUT, VMPRT2
FINDS2 --→ VEQUAL, MINIM2, TLOUT, VMPRT2

refby: ISOLAT

comm: DETINF, FILTRT, SYSU1, DETXBI, DCIDEI, SYNC, HFCOM, MCONCO, JUMPCM, NAMES.
(In addition, FINDSCMP/FINDS2 --→ MULTDT, SIGTAU & FINDS1 --→ SIG)

name: RECONF

func: Reconfigures the FTS for proper operation (if possible) after failures have been detected and isolated, and after sensors heal.

call: Call RECONF (Ihfail)

args: Ihfail -- integer in ; Heal/fail reconfiguration flag where Ihfail = 1 for failures and -1 for healings

refs: PNTINV, RCOV, SETISN, CLPSIO, TLOUT, IMTCG2
In addition,
FINDSCMP --→ NOISR, MATNL2
FINDS1/FINDS2 --→ MATNL3

refby: NAV

comm: DETINF, FILTRT, SYSU1, DETXBI, DCIDEI, SYNC, SYSXBO, INITVL, EKBFO, HEALCM, HFCOM, SYSX1, GBLEND, EKF1, ABRTCM.
In addition,
FINDSCMP --→ MULTDT
FINDS1 --→ SYSYW1, SIG
FINDS2 --→ MULTDT, SYSYW1, SIGTAU

name: CLPSIO

func: Used to collapse (or expand) the NFF and its associated data structures due to a single failure (or healing) of a measurement sensor. This routine is not called when an input sensor is involved.

1) If Iclps <0 (i.e., collapse NFF)
    * set RF1 (icmd) = 0
    * reset PF1 and PBFO by calling subroutine RCOV
    * decrement NY, NYF
    * update INOYP, INORYP, INOYPI
    * if meas. sensor bias is estimated, collapse bias portion of filter by calling subroutine CLPSBE
2) If Iclps >0 (i.e., expand NFF)
    * call NOISR to set RF1
    * increment NY, NYF
    * update INDYP, INORYP, INOYPI
    * correct partial derivative of h w.r.t. XF1, i.e., HP1 by calling UPDPH

call: Call CLPSIO (Iclps, Isns, Ireplc)

args: Iclps -- integer in ; flag used to control collapse/expansion of NFF where Iclps = 1 ==> collapse & Iclps = 1 ==> expand
Isns -- integer in ; absolute index of sensor (from Table 6.5)
Ireplc -- integer in ; replication of the sensor (1 or 2)

refs: RCOV, PNTINV, IMTCG2, CLPSBE
In addition, FINDSCMP --→ NOISR, UPDPH & FINDS 2 --→ UPDPH

refby: RECONF

comm: SYSXBO, SYSU1, SYSYW1, DETXBI, DETINF, INITVL, SYSX1
In addition, FINDS1 --→ FILTRT, SIG & FINDS2 --→ FILTRT, SIGTAU

name: NOISR (only in FINDSCMP)
func: Resets the measurement noise covariance terms in the NFF for a given sensor type and replication
call: Call NOISR (Isns, Ireplc, Imul)
args: Isns -- integer in ; absolute index of sensor (from Table 6.5)
Ireplc -- integer in ; not used
Imul -- integer in ; flag to use higher noise covariance when collapsing the IMU portion of filter. (default value = 1, value = 2 when IMU is involved)
refs: None
refby: RECONF, CLPSIO
comm: FILTRT, SYSYW1, SIGTAU, SYSU1

name: CLPSBE
func: Responsible for resetting the bias estimator portion of the NFF such that a single bias can be added or deleted
1) calls ADJTBP to determine IBkey and IYkey and to adjust the bias pointer vector INOBP, as well as NXB, NUB, NYB, NUB1, and NB
2) if kflag = -1 (i.e., collapse the bias estimator)
   (a) the IBkey row and column of the bias filter error covariance PBF0, is deleted.
   (b) the IBkey column of the bias filter blender gain, VB0 is deleted
   (c) the IBkey row of the bias estimation vector, XBF0, is deleted
3) if kflag ≠ -1 (i.e., expand the bias estimation)
   (a) PBF0 is expanded about the IBkey row and column, and they are zeroed out
   (b) the initial bias filter error covariance is loaded into the appropriate diagonal element s.t. PBF0 (IBkey) = PBF0I (Ibias) **2
   (c) VB0 is expanded about the IBkey column, and it is zeroed out.
   (d) XBF0 is expanded about the IBkey column, and zeroed out.
call: Call CLPSBE (kflag, Ibias)
args: kflag -- integer in ; flag to collapse/expand the bias filter
Ibias -- integer in ; absolute index of bias type to be added or deleted. (from Table 6.5)
refs: FINDSCMP --→ ADJTBP, MATCG2
FINDS1/FINDS2 --→ PNTINV, IMTCG2, MATCG3
refby: CLPSIO
comm: SYSXB0, EKBF0, INITVL, GBLEND
In addition, FINDS1/FINDS2 --→ SYSX1, DETXBI, CMPSTF, SYSU1, SYSYW1

name: ADJTBP (only in FINDSCMP)
func: Increments or decrements various vectors/scalars used by CLPSBE and the bias filter, when adding or deleting biases in the estimator
call: Call ADJTBP (Iflag, Index, Irkey, Iykey)
args: Iflag -- integer in ; flag indicating addition/deletion of bias (1 ==> add, -1 ==> delete)
Index -- integer in ; absolute index to sensor type of bias to be added or deleted (from Table 6.5)
Irkey -- integer out ; pointer to index in reduced bias set

Iykey -- integer out ; pointer to output type which corresponds to bias referred to by index. (If bias is an input bias, iykey = 0) Table 6.2

refs: PNTINV, IMTCG2
refby: CLPSBE
comm: SYSX1, DETXBI, CMPSTF, SYSXBO, SYSU1, SYSYW1

name: RCOV
func: Resets the NFF estimation error covariances once a failure has been detected and isolated. In particular, it sets,

$PF1 = PF1 + VBI * VBI^T + (XMI * XMI + 1.0/PMI)$

if PBFO > PBFOI ---→ PBFO = PBFOI, XBFO = 0

call: Call RCOV (Vi, Xmi, Pmi, Icmd)
args: Vi -- real in ; blender gain for i'th detector (i ≡ Table 6.6)
Xmi -- real in ; estimate of i'th failure level (i ≡ Table 6.6)
Pmi -- real in ; information matrix for i'th failure (i ≡ Table 6.6)
Icmd -- integer in ; absolute sensor type of failed sensor (Table 6.5)
refs: FINDSCMP ---→ VMPRT, MTNL2
FINDS1/FINDS2 ---→ VMPRT2, MATNL3
refby: CLPSIO, RECONF
comm: SYSXBO, EKBFO, EKF1, CMPSTF, SYSX1, INITVL
In addition, FINDSCMP ---→ MAIN1

name: HEALR
func: Manages the operation of the healer logic. Primary function is to maintain all sensor failed by the FDI logic and determine if they have healed or recovered. Healer decisions are made ONLY at the end of a healer decision window (which in our algorithms is set to be 3 seconds). In FINDSCMP/FINDS1, special logic is employed in order to force the IMU's to heal in a coordinated fashion.

HEALR is operated by computing the running sum, Xsum, of (Xwork-Xfail) over the healer window of length Kmxhlr (3 seconds). The value of the sum is reset to zero at the start of a new healer window; a new healer window is started whenever a new sensor is failed by the FDI logic. Xwork and Xfail are defined as follows:

* for input sensors:
  Xwork = measurement from a currently active replicated sensor of the same type as the failed one
  Xfail = measurement from the failed sensor
* for output sensors:
  Xwork = estimate of the observation obtained from the NFF
  Xfail = measurement from the failed sensor.

call: Call HEALR
args: None
refs: BUBBL2, TLOUT
In addition, FINDSCMP ---→ LRTHLR
refby: NAV
comm: SYNC, SYSU1, HEALCM, HFCOM, EKF1, YOBSRV, JUMPCM, NAMES, LOCHEA
In addition,
FINDSCMP ---→ AGOUT, SOUT, MLOUT, RGOUT, LAOUT, PSIR
FINDS1 ---→ AGOUT, RGOUT, PSIR
FINDS2 ---→ ASOUT, MLOUT, LAOUT

name: <u>LRTHLR</u> (only in FINDSCMP ; this routine is integrated into HEALR in FINDS1/FINDS2)

func: Performs a likelihood ratio test to determine if a sensor has healed at the end of a healer window. The test is performed as follows:

1) a maximum likelihood estimate of the normal operational bias is computed as, Best = Xsum/Length, where Xsum is the running sum from HEALR and Length is the number of samples in the window. The estimate is limited by:
   if Best > Bthrsh , Best = Bthrsh
   if Best < -Bthrsh , Best = Bthrsh

where Bthrsh is the largest expected bias level for this sensor type (set in INITG)

2) a maximum likelihood estimate for a failure level is computed as, Fest = Xsum/Length, which is then limited by:
   if Fest > 0 & Fest < Fthrsh , Fest = Fthrsh
   if Fest < 0 & Fest > -Fthrsh , Fest = -Fthrsh

where Fthrsh is the smallest expected failure level for this sensor type (set in INITG).

3) a decision function is evaluated as,
   Xtmp = 2.0 * (Fest -Best) * Xsum + Length * (Best **2 + Fest **2)

4) the value of the decision function is compared to a decision threshold, Dthrsh, (set in INITG), and if Xtmp < Dthrsh the sensor is declared "healed."

call: Call LRTHLR (Xsum, J)

args: Xsum -- real in ; sum of (Xwork -Xfail) over healer window
J -- integer in ; absolute index of failed sensor (refer Table 6.5)

refs: None

refby: HEALR

comm: HEALCM

name: <u>TLOUT</u>

func: Prints a coded message corresponding to an \`event\` and the status of the NFF estimates in the time-line file.

call: Call TLOUT (Msg, Imsg1, Imsg2)

args: Msg -- integer in ; message number corresponding to specific events.
Imsg1, Imsg2 -- integers in ; message qualifiers

refs: None

refby: FINDS/FINDS1/FINDS2, DECIDE, RECONF, HEALR

comm: MCONCO, SYNC, EKF1, EKBFO

<u>DESCRIPTION OF LIBRARY (MATRIX/VECTOR) ROUTINES</u>

name: <u>VMPRT/VMPRT2</u>

func: Prints out vectors or diagonals of matrices

call: Call VMPRT (X, Nr, Nc, Name) ←-- FINDSCMP
Call VMPRT2 (X, Nr, Nc, Name, Ndim1) ←-- FINDS1/FINDS2

args: X -- real in ; vector or matrix to be printed
Nr -- integer in ; row size of vector/matrix
Nc -- integer in ; column
Name -- character in ; character label to be printed
Ndim1 -- integer in ; only in exact dimensioned FINDS1 & FINDS2 -- max. row dimension of X in calling routine

refs: None
refby: DECIDE, RCOV
comm: None

name: BUBBL2
func: Performs a bubble sort on an array of integers where the final ordering is smallest to largest, i.e., increasing in value
call: Call BUBBL2 (Na, n)
args: Na -- integer in out ; array of integers to be sorted
n -- integer in ; length of array Na
refs: None
refby: INITG, HEALR
comm: None

name: MAT1A/PMAXB
func: Forms the matrix product Z = XY. No sparseness tests are performed and Z, Y can start at same core locations. PMAXB assumes that X, Y, Z are exact-dimensioned in the calling routine while MAT1A assumes a general `NDIM` row dimension for all matrices.
call: Call MAT1A/PMAXB (n1, n2, n3, X, Y, Z)
args: n1 -- integer in ; row dimension of X, Z
n2 -- integer in ; col. length of X, row length of Y
n3 -- integer in ; col. length of Y, Z
X -- real in ; input matrix (n1, N2)
Y -- real in ; input matrix (n2, n3)
Z -- real out ; output matrix (n1, n3)
refs: None
refby: EKFN1, BIASF, BLEND, ISOLAT, MAT3/PMABAT
comm: MAT1A --→ MAIN1
PMAXB --→ None

name: MAT2/PMABT2
func: Forms the matrix product $Z = XY^T$ where Z is symmetric. No sparseness tests are done and Z, Y can start at same core locations. PMABT2 assumes that X, Y, Z are exact-dimensioned in the calling routine while MAT2 assumes an `NDIM` row dimension for all matrices.
call: Call MAT2/PMABT2 (n1, n2, X, Y, Z)
args: n1 -- integer in ; row dimension of X, Y and col. length of Z
n2 -- integer in ; row dimension of X, Y
X -- real in ; input matrix (n1, n2)
Y -- real in ; input matrix (n1, n2)
Z -- real out ; output matrix (n1, n2)
refs: None
refby: EKFN1, ISOLAT
comm: MAT2 --→ MAIN1
PMABT2 --→ None

name: MAT3/PMABAT
func: Forms the symmetric matrix product $Z = X\ Y\ X^T$ where Y is symmetric, and no sparseness tests are done. PMABAT assumes that X, Y, Z are exact-dimensioned in the calling routine while MAT3 assumes an `NDIM` row dimension for all matrices.
call: Call MAT3 (n1, n2, X, Y, Z) ←-- FINDSCMP

Call PMABAT (n1, X, Y, Z) ←-- FINDS1/FINDS2 (assumes n1 = n2)

args: n1 --→ integer in ; row length of X, Z and col. length of Z
n2 --→ integer in ; row length of Y and col. length of X, Y
X --→ real in ; input matrix (n1, n2)
Y --→ real in ; input (symmetric) matrix (n2, n2)
Z --→ real out ; output (symmetric) matrix (n1, n1)

refs: MAT1A/PMAXB

refby: EKFN1

comm: MAT3 --→ MAIN1
PMABAT --→ None

name: MAT3B/PMVTAV

func: Forms a scalar output from the symmetric vector product $Z = V^T Y V$, where Y is symmetric and no sparseness tests are done. PMVTAV assumes that Y is exact-dimensioned in the calling routine while MAT3B assumes an `NDIM` row dimension for Y.

call: Call MAT3B/PMTAV (n1, V, Y, SOUT)

args: n1 -- integer in ; dimension of vector V and row/col. length of matrix Y
V -- real in ; input vector (n1, 1)
Y -- real in ; input (symmetric) matrix (n1, n1)
SOUT -- real out ; scalar output

refs: None

refby: DET01, DET05, DET10, ISOLAT, LRT

comm: MAT3B --→ MAIN1
PMVTAV --→ None

name: MATXYT/PMABT

func: Forms the matrix product $Z = X Y^T$, no sparseness test on Y PMABT assumes that X, Y, Z are exact-dimensioned in the calling routine while MATXYT assumes an `NDIM` row dimension on all matrices.

call: Call MATXYT/PMABT (n1, n2, n3, X, Y, Z)

args: n1 -- integer in ; row dimension of X, Z
n2 -- integer in ; col. length of X, Y
n3 -- integer in ; row length of Y, col. length of Z
X -- real in ; input matrix (n1, n2)
Y -- real in ; input matrix (n3, n2)
Z -- real out ; output matrix (n1, n3)

refs: None

refby: BIASF, ISOLAT

comm: MATXYT --→ MAIN1
PMABT --→ None

name: MEQUAL/PMBEA

func: Sets a matrix Y equal to a matrix X, Y = X
PMBEA assumes that X, Y are exact-dimensioned in the calling routine while MEQUAL assumes an `NDIM` row dimension for X, Y.

call: Call MEQUAL/PMBEA (n1, n2, X, Y)

args: n1 -- integer in ; row length of X, Y
n2 -- integer in ; col. length of X, Y
X -- real in ; input matrix (n1, n2)
Y -- real out ; output in (n1, n2)

refs: None

refby: BIASF, DET01, DET05, DET10, ISOLAT
comm: MEQUAL --→ MAIN1
PMBEA --→ None

name: TRANS2 (only in FINDSCMP)
func: Transpose a matrix, XPOSE = X^T. Assumes `NDIM` row dimension for all matrices.
call: Call TRANS2 (n1, n2, X, XPOSE)
args: n1 -- integer in ; row length of X, col. length of XPOSE
n2 -- integer in ; col. length of X, row length of XPOSE
X -- real in ; input matrix (n1, n2)
XPOSE -- real in ; output matrix (n2, n1)
refs: None
refby: ISOLAT
comm: MAIN1

name: PDMINV/PD3NV1/PD4NV1
func: Special matrix inverse routine for positive, symmetric, semi-definite matrices; uses Cholesky L-u decomposition as an intermediate step. PD3NV1 is the special form of PDMINV for 3rd order matrices, used in FINDS1 and PD4NV1 inverts 4th order matrices in FINDS2.
call: Call PDMINV (n, A, Ainv)
Call PD3NV1/PD4NV1 (A, Ainv)
args: n -- integer in ; order of matrix to be inverted
A -- real in ; input matrix (n, n)
Aimv -- real out ; output matrix (n, n)
NOTE: PD3NV1 & PD4NV1 assume n = 3 & n = 4, respectively.
refs: None
refby: EKFN1, BIASF, ISOLAT
comm: None

name: MINIM2/MINIM3
func: Searches a vector and determines the minimum value and its corresponding location. Only those elements of the vector are checked which have a corresponding non-zero element in the other input vector.
call: Call MINIM2 (Imactv, V, Npts, Vmin, Nmin)
Call MINIM3 (Imactv, V, Npts. Nmin)
args: Imactv -- integer in ; input vector (Npts, 1) with 0 or 1 entries corresponding to which entries in V to be checked.
V -- real in ; input vector (Npts, 1) to be searched
Npts -- integer in ; length of V (i.e., # of elements to be searched)
Vmin -- real out ; value of minimum element in V (not in MINIM3 which outputs only the location)
Nmin -- integer out; location of the minimum element in V
refs: None
refby: DECIDE
comm: None

name: PNTINV

func: Searches a pointer vector for particular entry. The pointer vector is an integer array with monotonically increasing elements. It will show how a collapsed vector's elements relate to a standard vector, i.e., given the absolute index, this routine reruns the active index in the reduced vector.

call: Call PNTINV (isns, Ipoint, n, index)

args: isns -- integer in ; value searched for in Ipoint (usually in absolute index)

Ipoint -- integer in ; pointer vector to be searched

n - integer in ; length of Ipoint

index -- integer out ; index in Ipoint where isns was found. If isns was not found, index < 0

refs: None

refby: RECONF, CLPSIO, ADJTBP

comm: None

name: IMTCG2

func: To add or delete a row in an integer matrix or vector, or to add or delete a column in a matrix. I a row or column is added, its elements are set to zero.

args: jflag -- integer in ; operation flag where:
1 --→ add row, 2 --→ add column
-1 --→ delete row, -2 --→ delete column

index -- integer in ; pointer to row/column to be added or deleted

IY -- integer in out ; matrix whose `index` row/column is to be added or deleted

nr -- integer in out ; # of rows of Y (incremented or decremented)

nc -- integer in out ; # of columns of Y (incremented or decremented) (not used in FINDS1/FINDS2)

refs: None

refby: RECONF, CLPSIO, ADJTBP, (CLPSBE)

comm: FINDSCMP --→ MAIN1
FINDS1/FINDS2 --→ None (as no column operations are performed)

name: MATCG2/MATCG3

func: To add or delete a row/column in a `real` matrix or vector. If a row or column is added, its elements are set to zero. MATCG3 assumes that the matrix is exact-dimensioned in the calling routine while MATCG2 assumes an `NDIM` row dimension for the matrix.

call: Call MATCG2 (jflag, index, Y, nr, nc)
Call MATCG3 (jflag, index, Y, nr, nc, ndiml)

args: jflag -- integer in ; operation flag where:
1 --→ add row, 2 --→ add column
-1 --→ delete row, -2 --→ delete column

index -- integer in ; pointer to row/column to be added or deleted

Y -- real in out ; matrix whose `index` row/column is to be added or deleted

nr -- integer in out ; # of rows of Y (incremented or decremented)

nc -- integer in out ; # of columns of Y (incremented or decremented)

ndiml -- integer in ; maximum row dimension of Y in calling routine (used only in FINDS1/FINDS2 for exact-dimensioned matrices)

```
refs:    None
refby:   CLPSBE
comm:    MATCG2 --→ MAIN1
         MATCG3 --→ None


name:    MATNL2/MATNL3
func:    Initializes columns n1 through n2 of a matrix to zero. In addition,
         if a flag is set, rows n1 through n2 can be nulled out, as well.
         MATNL3 assumes that the matrix is exact-dimensioned in the calling
         routine while MATNL2 assumes an `NDIM` row dimension for all
         matrices.
call:    Call MATNL2 (X, n1, n2, ktrig, n3)
         Call MATNL3 (X, n1, n2, ktrig, n3, ndim1)
args:    X -- real in out  ;  matrix whose rows/columns have to be nulled
         n1 -- integer in  ;  first column/row to be nulled
         n2 -- integer in  ;  last column/row to be nulled
         ktrig -- integer in  ; operation flag:  0 --→ only columns
                                                ≠0 --→ rows & columns
         n3 -- integer in  ;  # of elements in any row to be nulled (used
                              because all column dimensions are exact).
         ndim1 -- integer in  ; maximum row dimension of X in calling
                                routine. (only in exact-dimensioned
                                FINDS1/FINDS2)
refs:    None
refby:   ISOLAT, RECONF, RCOV
comm:    MATNL2 --→ MAIN1
         MATNL3 --→ None


name:    MADD/PMAPB
func:    Adds two matrices as Z = X + Y. PMAPB (used in FINDS1/FINDS2 assumes
         that all matrices are exact-dimensioned in the calling routine while
         MADD (used in FINDSCOMP) assumes an `NDIM` row dimension for all
         matrices.
call:    Call MADD/PMAPB (n1, n2, X, Y, Z)
args:    n1 -- integer in  ;  row length of X, Y, Z
         n2 -- integer in  ;  column length of X, Y, Z
         X -- real in  ;  input matrix (n1, n2)
         Y -- real in  ;  input matrix (n1, n2)
         Z -- real out  ;  output matrix (n1, n2)
refs:    None
refby:   EKFN1, BIASF, BLEND, ISOLAT
comm:    MADD --→ MAIN1
         PMAPB --→


name:    MSUB/PMAMB
func:    Matrix subtraction as Z = X - Y. PMAMB (used in FINDS1/FINDS2)
         assumes that all matrices are exact-dimensioned in the calling
         routine while MSUB (used in FINDSCMP) assumes an `NDIM` row dimension
         for all matrices.
call:    Call MSUB/PMAMB (n1, n2, X, Y, Z)
args:    refer args. for MADD/PMAPB
refs:    None
refby:   BIASF, ISOLAT
comm:    MSUB --→ MAIN1
         PMAMB --→
```

name: <u>MATVEC/PMAXV</u>
func: Performs matrix vector multiplication as V2 = X*V1. PMAXV (used in FINDS1/FINDS2) assumes that the matrix X is exact-dimensioned in the calling routine while MATVEC (used in FINDSCMP) assumes an `NDIM` row dimension for all matrices.
call: Call MATVEC/PMAXV (n1, n2, X, V1, V2)
args: n1 -- integer in ; row length of X, length of V2
n2 -- integer in ; column length of X, length of V1
X -- real in ; input matrix (n1, n2)
V1 -- real in ; input vector (n2)
V2 -- real out ; output vector (n1)
refs: None
refby: BIASF, ISOLAT
comm: MATVEC --→ MAIN1
PMAXV --→ None

name: <u>MATVC2/PMAXV2</u>
func: Computes matrix-vector-product-sum as V3 = X*V1 + V2 (an extension of MATVEC/PMAXV)
call: Call MATVC2/PMAXV2 (n1, n2, X, V1, V2, V3)
args: same as MATVEC/PMAXV with exception of
V2 -- real in ; input vector (n1)
V3 -- real out ; output vector (n1)
refs: None
refby: BLEND
comm: MATVC2 --→ MAIN1
PMAXV2 --→ None

name: <u>VSCALE</u>
func: Performs vector scaling as V2 = s*V1
call: Call VSCALE (n1, stmp, V1, V2)
args: n1 -- integer in ; length of vectors V1, V2
stmp -- real in ; scale factor
V1 -- real in ; input vector to be scaled
V2 -- real out ; output vector
refs: None
refby: EKFN1, BIASF
comm: None

name: <u>VEQUAL</u>
func: Equates vectors as, V2 = V1
call: Call VEQUAL (n1, V1, V2)
args: n1 -- integer in ; length of V1, V2
V1 -- real in ; input vector
V2 -- real out ; output vector
refs: None
refby: INITG, ISOLAT, DECIDE
comm: None

name: <u>VADD</u>
func: Performs vector addition as V3 = V1 + V2
call: Call VADD (n1, V1, V2, V3)

```
args:    n1 -- integer in  ;  length of V1, V2, V3
         V1 -- real in  ;  input vector
         V2 -- real in  ;  input vector
         V3 -- in out  ;  output vector (result of addition)
refs:    None
refby:   ISOLAT
comm:    None


name:    VSUB
func:    Performs vector subtraction as, V3 = V1 - V2
call:    Call VSUB (n1, V1, V2, V3)
args:    same as VADD
refs:    None
refby:   BIASF, ISOLAT
comm:    None
```

8. COMMON BLOCK DESCRIPTION AND TABLES

This section contains a list of FINDS program variables as partitioned by various common blocks. Table 8.1 is a "short form" list of each common block in FINDS and the various subprograms which refer to it. Supporting Table 8.1 is a detailed description of the variables contained in each block.

TABLE 8.1

COMMON BLOCKS

Common Block	Referenced by Subprogram(s)
ABRTCM	FINDS (main), NAV, RECONF
AGOUT	READFL, INITXF, SUMOUT, RESCMP, HEALR
ASOUT	READFL, INITG, INITXF, SUMOUT, RESCMP, HEALR
BIASF	
BLNDWK	
BNSRT1	BNSAV1, BNSAV2, SORTER
BNSRT2	BNSAV2, BNSAV2, SORTER
BNSRT2	SORTER
BSFWFK	
CMPSTF	INITG, BLEND, UPDPH, ISOLAT, RCOV
CNTROL	FINDS (main), NAV, INITG, EKFNI, BIASF, DFT01, DFT05, DET10
DCIDEI	ISOLAT, DECIDE, RECONF
DETCOV	BIASF, ISOLAT
DETINF	INITG, BLEND, SFTISN, RESCMP, ISOLAT, BNSAV2, LKF, LRT, DECIDE, RECONF, CLPSIO
DETLC2	ISOLAT
DETLC3	ISOLAT, LKF, LRT
DETPRI	DET01, DET05, DET10
DETWRK	
DETXBI	INITG, SUMOUT, SETISN, ISOLAT, LKF, DECIDE, RECONF, CLPSIO
DETYBI	ISOLAT, LKF, LRT
DTCT01	DET01, BNSAVI
DTCT05	DET05, BNSAVI
DTCT10	DET10, BNSAVI
DTSYNC	NAV, RESCMP, ISOLAT, BNSAV2
EARTH	FINDS (main), SUMIN, GTOI
EKBFO	INITG, SUMIN, BIASF, BLEND, DET01, DET05, DET10, UPDH, ISOLAT, RECONF, ILOUT, CLPSBE, RCOV, BNSAV1, BNSAV2
EKFI	NAV, INITG, INITXF, GTO8, EKFNI, BIASF, BLEND, UPDB, UPDH, RESCMP, TLOUT, BNSAV2, ISOLAT, RECONF, RCOV, HEALR
EKFBIA	EKFNI, BIASF
EKFBLN	
EKFWRK	
EULER	UPDB
FILTIC	INITG

Common Block	Referenced by Subprogram(s)
FILTRT	INITG, INITXF, SUMIN, SUMOUT, GTOI, EKFNI, BIAS, BLEND, DET01, DET05, DET10, NOISR, RECONF, DECIDE, SETISN, RESCMP, ISOLAT
FLTIN	
GBLEND	BIASF, BLEND, RECONF, CLPSBE
GRVYTC	GTOI
GTOILC	GTOI
HEALCM	NAV, INITG, RECONF, HEALR, LRTHLR
HFCOM	NAV, DECIDE, RECONF, HEALR
IMLS	FINDS (main), GTOI
INITVL	INITG, ISOLAT, RECONF, CLPSIO, CLPSBE, RCOV
JUMPCM	NAV, EKFNI, BIASF, BLEND, DET01, DET10, DECIDE, HEALR
LAOUT	READFL, SUMIN, HEALR
LATLON	SUMIN, GTOI, BNSAV2
LCOM21	BNSAV1, BNSAV2,
LOCHEA	HEALR
LRTINV	BIASF, DET01, DET05, DET10
LRTMAX	FINDS (main), DET01, DET05, DET10
MAIN1	INITG, SUMIN, GTOI, BIASF, UPDB, UPDPH, ISOLAT, LKF, LRT, RCOV, BNSAV2
MAIN2	EKFNI, BIASF, BLEND, ISOLAT, BNSAV2
MCONCO	FINDS (main), READFL, INITG, INITXF, GTOI, DECIDE, BNSAV1, BNSAV2, TLOUT
MLOUT	READFL, INITXF, SUMOUT, RESCMP, HEALR
MSALL	FINDS (main), INITXF, UPDH, UPDPH,
MULTDT	ISOLAT, DECIDE, RECONF
NAMES	READFL, DECIDE, HEALR
PSIR	FINDS (main), INITXF, SUMIN, SUMOUT, GTOI, RESCMP, HEALR, BNSAV2
PQRDEG	GTOI
RDLOCL	READFL
RGOUT	READFL, SUMIN, GTOI, HEALR
SIGTAU	FINDS (main), INITG, DECIDE, NOISR
SUMLOC	SUMIN
SYNC	FINDS (main), READFL, NAV, INITG, SUMIN, DET01, DET05, DET10, UPDB, DECIDE, RECONF, HEALR, BNSAVI, TLOUT
SYSUI	INITG, SUMIN, GTOI, EKFNI, BIASF, BLEND, DET01, DET05, DET10, SETISN, UPDB, UPDH, BNSAV2, HEALR, ADJTBP, NOISR, CLPSIO, RECONF, DECIDE, UPDPH, RESCMP, ISOLAT
SYSXI	INITG, EKFNI, BIASF, BLEND, UPDB, UPDH, UPDPH, ISOLAT, RECONF, CLPSIO, ADJTBP, RCOV
SYSXBO	NAV, INITG, EKFNI, BIASF, BLEND, UPDB, UPDH, UPDPH, ISOLAT, RECONF, CLPSIO, CLPSBE, ADJTBP, RCOV, BNSAV2
SYSYBO	EKFNI, BIASF, RECONF,
SYSYWI	INITG, SUMOUT, EKFNI, BIASF, BLEND, DET01, DET05, DET10, UPDH, UPDPH, ISOLAT, CLPSIO, NOISR, ADJTBP, BNSAV2
TRBER	SUMIN, GTOI, UPDB,
TSTORE	EKFNI, BIASF, BLEND, ISOLAT
UPDQLC	
YOBSRV	INITG, SUMOUT, BIASF, UPDH, UPDPH, RESCMP, ISOLAT, HEALR, BNSAVI

II. DESCRIPTION OF COMMON BLOCKS

NOTE: (1) All vector/matrix dimensions are specified here in three separate parentheses corresponding to their use in FINDSCMP, FINDS1 and FINDS2, respectively. A single parentheses implies that all three versions use the same dimensions.

(2) Notation used is as follows: cont ==> contains (i.e., brief description of common block)
vars ==> variables contained in common block

name: ABRTCM
cont: System status flag
vars: iabort -- integer, unitless ; program abort flag which is activated (i.e., set from 0 to 1) when too many sensors are failed by the FDI logic and filter cannot operate with the remaining sensor complement.
refby: FINDS/FINDS1/FINDS2, RECONF

name: AGOUT (not in FINDS2)
cont: IMU sensor measurements from flight data in program units
vars: Phim -- real, radians, (2) (2) (-) ; dual replicated IMU roll measurements
Them -- real, radians, (2) (2) (-) ; dual replicated IMU pitch measurements
Psim -- real, radians, (2) (2) (-) ; dual replicated IMU yaw measurements (w.r.t. North)
refby: READFL, INITXF, SUMOUT, RESCMP, HEALR

name: ASOUT (not in FINDS1)
cont: IAS measurements from flight data in program units.
vars: Airsm -- real, m/s, (2) (-) (2) ; dual replicated airspeed measurements
refby: READFL, INITXF, SUMOUT, RESCMP, HEALR

name: BLNDWK (only in FINDS2)
cont: Temporary working variable(s) in subroutine BLEND
vars: Vtmpl -- real, mixed units, (-) (-) (8) ; temp. vector used in propagation
refby: BLEND

name: BSFWRK (not in FINDSCMP)
cont: Local working variables/arrays in subroutine BIASF
vars: Cbf0 -- real, mixed units, (-) (3,3) (4,3) ; bias filter observation matrix
Com2 -- real, mixed units, (-) (-) (8,8) ; temporary local matrix
Tmp1 -- real, mixed units, (-) (-) (8,3) ; temporary local matrix
Tmp2 -- real, mixed units, (-) (-) (8,3) ; temporary local matrix
Tmp3 -- real, mixed units, (-) (-) (3,4) ; temporary local matrix
Tmp4 -- real, mixed units, (-) (-) (4,4) ; temporary local matrix
Tmp5 -- real, mixed units, (-) (-) (3,3) ; temporary local matrix

refby: BIASF

name: <u>CMPSTF</u>
cont: Quantities associated with composite NFF (bias free + bias)
vars: nxb -- integer, unitless ; total states + bias states in NFF value = (17) (6) (11)
Pxfl -- real, mixed, (17,17) (6,6) (11,11) ; combined NFF estimation error covariance
refby: INITG, BLEND, UPDPH, ISOLAT, CLPSBE/ADJTBP, RCOV

name: <u>CNTROL</u>
cont: Option flag to activate/deactivate FDI logic
vars: icntrl -- boolean, unitless ; false ==> run NFF only
true ==> run FDI portion of algorithm also
refby: FINDS1/FINDS1/FINDS2, NAV, INITG, EKFN1, BIASF, DET01, DET05, DET10

name: <u>DCIDEI</u>
cont: Quantities relevant to the LR computations and the decision logic
vars: Priori -- real, unitless, (20) (9) (11) ; vector of log of prior probabilities of failure -- one for each sensor, ordered by replicated sensor index of Table 6.6 but assumes dual MLS replication.
Alamda -- real, unitless, (20) (9) (11) ; vector of log - likelihood of sensor failing -- one for each sensor, ordered by replicated sensor index of Table 6.6 but assumes dual MLS replication.
refby: ISOLAT, DECIDE, RECONF

name: <u>DETCOV</u>
cont: Quantity needed in isolation routine to form total covariance
vars: Afvb -- real, unitless, (17,6) (3,3) (8,3) ; intermediate storage matrix which saves the computation AF1*VB0 + BF1
refby: BIASF, ISOLAT

name: DETINF
cont: Information pertinent to the bank of first order filters in ISOLAT
vars: nft -- integer, unitless ; total # of replicated sensors (considered for FDI) value = (17) (9) (8)
nyf -- integer, unitless ; current # of replicated measurement sensors value = (11) (6) (5) from Table 6.7

	Inoryp -- integer, unitless, (17) (6) (11) ;	pointer vector to measurement sensor type
	Icntsn -- integer, unitless, (20) (9) (11) ;	determines if a particular sensor type/replication is being used and which element of the input/meas. vector it corresponds to. Null entry implies an inactive sensor. (Table 6.6)
	Resboc -- real, mixed, (14,10) (6,10) (8,10) ;	expanded residual vector fram the NFF saved over the last 10 iterations
refby:	INITG, BLEND, SETISN, RESCMP, ISOLAT, LKF, LRT, DECIDE, RECONF, CLPSIO	

name:	DETLC2 (not in FINDS1)	
cont:	Variables relevant to multiple MLS sensor failures	
vars:	Dobs - real, mixed, (17,6) (-) (5,6) ;	observation matrix to generate dual failure conditioned residuals
	Best -- real, mixed, (6) ;	estimated magnitude of multiple replicated MLS failure
refby:	ISOLAT	

name:	DETLC3	
cont:	Quantities local to the isolation logic which are temporarily stored	
vars:	Detinv -- real, mixed, (17,14) (6,6) (11,5) ;	inverse of expanded innovations covariance
	Hpaf -- real, mixed, (17,11) (-) (4,8) ;	computed HP1*AF1
	Hpbf -- real, mixed, (17,6) (6,3) (11,3) ;	computed HP1*BF1
	Augm -- real, mixed, (17,6) (6,3) (11,3) ;	intermediate augmented matrix
	Hbpd -- real, mixed, (17,6) (6,3) 11,3) ;	computed HP1*BF1 + D
	Bmghb -- real, mixed, (17,6) (6,3) (11,3) ;	BF1 - GAINKX*HP1*BF1
refby:	ISOLAT, LKF, LRT	

name:	DETPRI	
cont:	Flag to check if a failure has already been detected in current iteration ; hierarchy of detectors is DET01, DET10, DET05	
vars:	idfail -- integer, unitless ;	set to 1 by any detector which flags a sensor failure -- remaining detectors will be deactivated during current iteration (default value = 0)
refby:	DET01, DET05, DET10	

name: DETWRK (not in FINDSCMP)
cont: Local working arrays in subroutine ISOLAT
vars: Vtmp1 -- real, mixed, (-) (6) (11) ; temporary working vector
Vtmp2 -- real, mixed, (-) (6) (11) ; temporary working vector
Tmp1 -- real, mixed, (-) (3,3) (8,8) ; temporary working matrix
Tmp2 -- real, mixed, (-) (6,6) (5,11) ; temporary working matrix
Tmp3 -- real, mixed, (-) (6,6) (5,5) ; temporary working matrix
Tmp4 -- real, mixed (-) (-) (5,5) ; temporary working matrix
Com2 -- real, mixed, (-) (-) (8,3) ; temporary working matrix
Hpic -- real, mixed, (-) (6,6) (5,11) ; composite observation matrix
Gnkxd -- real, mixed, (-) (6,3) (11,4) ; augmented NFF gain matrix [GAINKX/GAINBO]^T
refby: ISOLAT

name: DETXBI
cont: Quantities associated with the sensor failure isolation & estimation logic
vars: nfmax -- integer, unitless ; maximum possible # of sensor types to be considered (has value = 13, 6, 7)
nymax -- integer, unitless ; maximum possible # of measurement sensor types to be considered (has value = 7, 3, 4)
xbfi -- real, mixed, (20) (9) (11) ; vector of current failure level estimates -- one for each type & replication using absolute indexing (Table 6.6)
Pbfi -- real, mixed, (20) (9) (11) ; vector of estimation information for each estimated failure (ordered as per Table 6.6)
Vbi -- real, mixed, (17,13) (6,6) (11,7) ; matrix of blender gain vectors
refby: INITG, SUMOUT, SETISN, ISOLAT, LKF, DECIDE, RECONF, CLPSIO, ADJTBP

name: DETYBI
cont: Observation matrices and compensated residual vectors for the bank of filters in the isolation logic
vars: Resbi -- real, mixed, (17,20) (6,9) (11,11) ; matrix of failure compensated residuals vectors - cols. are ordered by replicated sensor index (Table 6.6)
Cbfi -- real, mixed, (17,13) (6,6) (11,7) ; observation matrix where each col. is an observations vector for a filter. Cols. are ordered by replicated sensor index (Table 6.6)
refby: ISOLAT, LKF

name: <u>DTCT01</u>
cont: Quantities associated with the detector of window length 1 sample
vars: vlrt01 -- real, unitless ; Chi-square test failure likelihood ratio
Rti01 -- real, mixed, (17,7) (3,3) (4,4) ; NFF innovations inverse matrix compensated for residual window length of 1 sample
refby: DET01

name: <u>DTCT05</u>
cont: Quantities associated with the detector of window length 5 samples
vars: vlrt05 -- real, unitless ; Chi-square test failure likelihood ratio
Ravg05 -- real, mixed, (7) (3) (4) ; five sample moving window average of NFF residuals RESB0
Rsav05 -- real, mixed, (7,5) (3,5) (4,5) ; saved RESB0 over last five iterations (moving window)
Rti05 -- real, mixed, (17,7) (3,3) (4,4) ; NFF innovations inverse matrix compensated for residual window length of 5 samples
refby: DET05

name: <u>DTCT10</u>
cont: Quantities associated with the detector of window length 10 samples
vars: vlrt10 -- real, unitless ; Chi-square test failure likelihood ratio
Ravg10 -- real, mixed, (7) (3) (4) ; ten sample moving window average of NFF residuals RESB0
Rsav10 -- real, mixed, (7,10) (3,10) (4,10) ; saved RESB0 over last ten iterations (moving window)
Rti10 -- real, mixed, (17,7) (3,3) (4,4) ; NFF innovations inverse matrix compensated for residual window length of 5 samples
refby: DET10

name: <u>DTSYNC</u>
cont: Pointer to current location in saved array of NFF expanded residuals
vars: icurnt -- integer, unitless ; [1,10] location in saved RESB0C -- used in ISOLAT to go back either 5 or 10 iterations and run isolation logic
refby: NAV, RESCMP, ISOLAT

name: EARTH (not in FINDS2)
cont: Quantities associated with earth's rotation -- used in GTOI to compute a/c latitude, longitude and rate gyro compensation terms
vars: omegt -- real, radians ; computed WE * TIME to give angular change between I-frame and E-frame
sinet -- real, unitless ; sine of omegt
comet -- real, unitless ; cosine of omegt
re -- real, meters ; radius of earth
we -- real, rad/s ; earth rotation rate
refby: FINDS/FINDS1, SUMIN, GTOI

name: EKBF0
cont: Arrays used in the bias filter portion of the NFF
vars: Xbf0 -- real, mixed, (17) (3) (3) ; vector of current normal operating bias estimates (Table 6.3)
Resb0 -- real, mixed, (7) (3) (4) ; vector of NFF residuals (Table 6.2)
Gainb0 -- real, mixed, (17,7) (3,3) (3,4) ; Kalman gain for bias filter
Pbf0 -- real, mixed, (17,6) (3,3) (3,3) ; bias filter estimation error covariance
refby: INITG, SUMIN, BIASF, BLEND, DET01, DET05, DET10, UPDH, ISOLAT, RECONF, CLPSBE, RCOV, TLOUT

name: EKF1
cont: Arrays used in the bias free portion of the NFF
vars: Xf1 -- real, mixed, (11) (3) (8) ; vector of current NFF state estimates (Table 6.1)
Hxkp1 -- real, mixed, (7) (3) (4) ; vector of NFF observations (Table 6.2)
Gainkx -- real, mixed, (17,7) (3,3) (8,4) ; Kalman gain for EKF (bias and bias-free)
Pf1 -- real, mixed, (17,11) (3,3) (8,8) ; bias free filter estimation error covariance
refby: NAV, INITG, INITXF, GTOI, EKFN1, BIASF, BLEND, UPDB, UPDH, UPDPH, RESCMP, ISOLAT, RECONF, RCOV, HEALR, TLOUT

name: EKFBIA
cont: arrays common to the bias and bias-free filters
vars: Ximgh -- real, mixed, (17,11) (3,3) (8,8) ; saved computed I-GAIN *HP1
Tmp1 -- real, mixed, (-) (3,3) (-) ; temporary working matrix
Tmp2 -- real, mixed, (-) (3,3) (-) ; temporary working matrix
Rbf0 -- real, mixed, (-) (-) (4,4) ; saved HP1 * PF2 * $HP1^T$ + R computed in EKFN1 and used also in BIASF
refby: EKFN1, BIASF

name: EKFBLN (only in FINDS2)
cont: Working arrays common to subroutines EKFN1 & BLEND

```
vars:     Tmp3 -- real, mixed (-) (-) (8,4)  ;  temporary working matrix
refby:    EKFN1, BLEND


name:     EKFWRK (only in FINDS2)
cont:     Working arrays local to subroutine EKFN1
vars:     Tmp1 -- real, mixed, (-) (-) (8,8)  ;  temporary working matrix
          Tmp2 -- real, mixed, (-) (-) (8,8)  ;  temporary working matrix
          Gktmp -- real, mixed, (-) (-) (4,4)  ; intermediate gain matrix
                                                  calculation
refby:    EKFN1


name:     EULER
cont:     Sine/cosine values of a/c Euler angles
vars:     s1 -- real, unitless  ;  sine of roll attitutde
          c1 -- real, unitless  ;  cosine of roll attitude
          s2 -- real, unitless  ;  sine of pitch attitude
          c2 -- real, unitless  ;  cosine in pitch attitude
          t2 -- real, unitless  ;  tangent of pitch attitude
          s3 -- real, unitless  ;  sine of yaw attitude
          c3 -- real, unitless  ;  cosine of yaw attitude
refby:    UPDB


name:     FILTIC
cont:     Variables associated with NFF initial conditions
vars:     Sdpic -- real, mixed, (11) (3) (8)  ;  vector of s.d. of the
                                                 diagonal elements of the NFF
                                                 state initial estimation
                                                 error covariance.
refby:    INITG


name:     FILTRT
cont:     Pointing vectors used by NFF
vars:     mxrplf -- integer, unitless  ;  max. # sensor replications used in
                                          the NFF & FDI logic -- currently
                                          limited to 2.
          Ireplf -- integer, unitless, (13) (6) (7)  ;  vector of sensor
                                                        replications used by
                                                        the NFF (absolute
                                                        sensor indexing)
                                                        (Table 6.5)
          Inoutf -- integer, unitless, (17,2) (6,2) (7,2)  ;  matrix
                                                              indicating
                                                              status of all
                                                              sensors in the
                                                              NFF. Row index
                                                              corresponds to
                                                              absolute sensor
                                                              type and col.
                                                              index is
                                                              replication of
                                                              sensor.  1 ==>
                                                              active, -1 ==>
                                                              standby, 0 ==>
                                                              failed
```

refby: INITG, SUMIN, SUMOUT, GTOI, EKFNL, BIASF, BLEND, DET01, DET05, DET10, SETISN, ISOLAT, DECIDE, RECONF

name: FLTIN (only in FINDS2)
cont: Vector array of sensor flight data
vars: Readin -- real, mixed, (-) (-) (26) ; dual replicated sensor flight data.
refby: READFL, INITXF

name: GBLEND
cont: NFF blender gain matrix
vars: Vb0 -- real, mixed, (17,6) (3,3) (8,3) ; NFF blender gain
refby: BIASF, BLEND

name: GRVYTC (not in FINDS2)
cont: Arrays needed to compute gravity vector which is appended to the input vector UF1
vars: GRavlc -- real, m/s^2, (3) (3) (-) ; skew symmetric compensation terms for runway frame w.r.t. inertial frame
Tlcprt -- real, unitless, (3) (3) (-) ;
refby: GTOI

name: GTOILC (not in FINS2)
cont: Saved local variables in subroutine GTOI
vars: aloni -- real, radians ; constant longitude offset
alati -- real, radians ; constant latitude offset
ticp1 -- real, unitless ; constant term in transformation matrix Tic
ticp2 -- real, unitless ; constant term in transformation matrix Tic
ticp3 -- real, unitless ; constant term in transformation matrix Tic
ticp4 -- real, unitless ; constant term in transformation matrix Tic
ticp5 -- real, unitless ; constant term in transformation matrix Tic.
ticp6 -- real, unitless ; constant term in transformation matrix Tic
refby: GTOI

name: HEALCM
cont: Quantities used by the healer logic
vars: kcthlr -- integer, unitless ; running count of elapsed samples since start of current healer window; value = [1, 60]
kmxhlr -- integer, unitless ; total # of samples in (i.e., length of) healer window; value = 60
confbd -- real, unitless ; log of initial confidence bound (1/19) for the healer test

bthrsh -- real, mixed, (13) (6) (7) ; vector of largest expected normal operating biases for each sensor type -- absolute sensor index, Table 6.5

Fthrsh -- real, mixed, (13) (6) (7) ; vector of smallest expected failure levels for each sensor type (Table 6.5)

Dthrsh -- real, mixed, (13) (6) (7) ; vector of decision thresholds to be applied to each healer process. Dthrsh (i) = 2*Confbd*Phealt (i)**2 where Phealt contains s.d. of expected noise to be used only by healers (Table 6.5)

refby: INITG, NAV, RECONF, HEALR, LRTHLR

name: <u>HFCOM</u>

cont: Quantities common to the healing/failure reconfiguration logic.

vars: nfail -- integer, unitless ; total # of sensors determined to be 'failed'

nnfail -- integer, unitless ; # of new failures, i.e., incremental # of sensors just detected as failed in current iteration

nhealm -- integer, unitless ; max. # of sensors which can heal in one instant (i.e., dimension of Ihealp)

nheal -- integer, unitless ; total # of sensors which the healer logic has declared healthy at the end of a healer window

Ifailt -- integer, unitless, (13) (6) (7) ; vector containing absolute sensor type for each failed sensor. (Table 6.5)Whenever a sensor fails, its absolute sensor type is added to Ifailt -- hence, this vector is ordered by relative time of occurrence of failure.

Ifailr -- integer, unitless, (13) (6) (7) ; vector containing replication # for each failed sensor -- ordered same as Ifailt

Ihealp -- integer, unitless, (10) (6) (7) ; vector containing list of failed sensors which have healed. The value of an element is the index in Ifailt/Ifailr of the healed sensor.

refby: NAV, RECONF, HEALR

name: IMLS (not in FINDS2)
cont: Quantities associated with earth rotation & thus on MLS frame rotation.
vars: rmagor -- real, m ; radius of earth added to mean sea level altitude of MLS frame origin
slat -- real, radians ; latitude of MLS frame origin
slon -- real, radians ; longitude of MLS frame origin
sinlac -- real, unitless ; sine of slat
coslac -- real, unitless ; cosine of slat
Wrws -- real, unitless, (9) (9) (-) ; skew symmetric form of angular vel. of runway w.r.t. inertial frame.
refby: FINDS/FINDS1, GTOI

name: INITVL
cont: Initial values for the NFF
vars: Inobps -- integer, unitless, (13) (6) (7) ; INOBPS=INOBP at start of run (showing which sensor biases are to be estimated) Table 6.5
Pbf0i -- real, mixed, (13) (6) (7) ; initial s.d. of bias estimation error (user units) Table 6.5
Pbfic -- real, mixed, (13) (6) (7) ; initial s.d. of isolator filters error information. (user units) (absolute sensor index) Table 6.5
refby: INITG, RECONF, CLPSIO, CLPSBE, RCOV

name: JUMPCM
cont: Variables for multi-frequency implementation of NFF.
vars: jmpcvx -- integer, unitless ; # of iterations after which bias free covariance has to be computed
jmpcvb -- integer, unitless ; # of iterations after which bias covariance has to be computed
jmpgnx -- integer, unitless ; # of iterations after which bias free gain has to be computed
jmpgnb -- integer, unitless ; # of iterations after which bias gain has to be computed
jiter -- integer, unitless ; running counter of iterations or elapsed time ticks
jmdcx -- integer, unitless ; mod (jiter, jmpcvx) = 0 ==> perform computations
jmdcb -- integer, unitless ; mod (jiter, jmpcvb) = 0 ==> perform computations
jmdgx -- integer, unitless ; mod (jiter, jmpgnx) = 0 ==> perform computations
jmdgb -- integer, unitless ; mod (jiter, jmpgnb) = 0 ==> perform computations
refby: NAV, EKFN1, BIASF, BLEND, DET01, DET05, DET10, DECIDE, HEALR

name: LAOUT (not in FINDS1)
cont: Replicated accelerometer sensor measurements from flight data
vars: Axm -- real, m/s^2, (2) (-) (2) ; dual longitudinal accelerometer meas.
Aym -- real, m/s^2, (2) (-) (2) ; dual lateral accelerometer meas.
Azm -- real, m/s^2, (2) (-) (2) ; dual vertical accelerometer meas.
refby: READFL, SUMIN, HEALR

name: LATLON (not in FINDS2)
cont: Information regarding a/c latitude and longitude
vars: alat -- real, radians ; current estimate of a/c latitude
alon -- real, radians ; current estimate of a/c longitude
alatd -- real, rad/s ; current estimat of rate of latitude change
alond -- real, rad/s ; current estimat of rate of longitude change
csalat -- real, unitless ; cosine of alat
snalat -- real, unitless ; sine of alat
refby: GTOI, SUMIN

name: LOCHEA
cont: Quantities local to subroutine HEALR
vars: nfaill -- integer, unitless ; local snapshot of `nfail`
Ifailp -- integer, unitless, (20) (9) (11) ; local snapshot of `Ifailt`
Xsum -- real, mixed, (20) (9) (11) ; running sum over healing window length of difference between failed sensor and "working" sensor.
Itest -- integer, unitless, (3) (3) (-) ; Local pointer vector for IMU healing logic
Itestp -- integer, unitless, (3,3) (3,3) (-) ; Local pointer to store which parts of the IMU have healed
Itest2 -- integer, unitless, (9) (9) (-) ; pointer vector to check that entire IMU heals as a unit
refby: HEALR

name: LRTINV
cont: Saved part of Kalman gain calculations from bias filter to be used by the detectors in the Chi-square test.
vars: Rtinv -- real, mixed, (17,7) (3,3) (4,4) ; saved [CBF0*PBF0*CBF0 + RBF0] ** -1
refby: BIASF, DET01, DET05, DET10

name: LRTMAX
cont: Maximum Chi-square test thresholds to trip detectors
vars: vmax01 -- real, unitless ; max. threshold to trip DET01
vmax05 -- real, unitless ; max. threshold to trip DET05
vmax10 -- real, unitless ; max. threshold to trip DET10
refby: DET01, DET05, DET10

name: MAIN1
cont: Provides common dimensioning information for all 2-dimensional arrays and a scratch array for temporary use by all routines.
vars: ndim -- integer, unitless ; common row dimension for all arrays, value = (17) (6) (11)
ndiml -- integer, unitless ; ndim +1
Dmfx -- real, temporary, (17,17) (6,6) (11,11) ; scratch area dimensioned `ndim x ndim`
refby: INITG, SUMIN, GTOI, BIASF, UPDB, UPDQ, UPDPM, ISOLAT, RCOV, VMPRT, MAT1A, MAT2, MAT3, MAT3B, MATXYT, MEQUAL, TRANS2, IMTCG2, MATCG2, MAINL2, MADD, MSUB, MATVEC, MATVC2

name: MAIN2
cont: Provides a temporary scratch area for use by all routines
vars: Com2 -- real, temporary, (17,17) (6,6) (11,11) ; scratch array dimensioned `ndim x ndim`
refby: EKFN1, BIASF, BLEND, ISOLAT

name: MCONCO
cont: Conversion factors from user units to program units & vice versa
vars: Radian -- real, unitless ; conversion factor from degrees to radians
Cnvrf -- real, unitless, (13) (-) (7) ; conversion factors from program units to user units for sensor signals -- absolute sensor index. Table 6.5 (not used in FINDS1 because all conversions are radians to degrees)
refby: FINDS/FINDS1/FINDS2, READFL, INITG, INITXF, GTOI, UPDQ, DECIDE, TLOUT

name: MLOUT (not in FIND21)
cont: Replicated MLS sensor measurements from flight data
vars: Azim -- real, radians, (2) (-) (2) ; dual azimuth measurements
Elem -- real, radians, (2) (-) (2) ; dual elevation measurements
Rngm -- real, radians, (2) (-) (2) ; dual range measurements
refby: READFL, INITXF, SUMOUT, RESCMP, HEALR

name: MLSALL (not in FINDS1)
cont: Information regarding MLS antenna locations.
vars: Xaz -- real, m, (3) (-) (3) ; location of azimuth/DME antenna in the runway frame
Xel -- real, m, (3) (-) (3) ; location of elevation/DME antenna in the runway frame
x0 -- real, m ; x-location of elev. antenna in MLS frame
x0 -- real, m ; y-location of elev. antenna in MLS frame
z0 -- real, m ; altitude offset between azimuth & elev. antennae
refby: FINDS/FINDS2, UPDH, UPDPH

name: MULTDT (not in FINDS1)
cont: Quantities used in detecting multiple simultaneous failures.
vars: Priorj -- real, mixed, (3) (-) (3) ; vector of log. of the prior probability of more than one MLS sensor of the same type to fail in the same instant (common mode failure). (ordered MLS azimuth, elevation, range)
Alamdj -- real, mixed, (3) (-) (3) ; vector of log-likelihood of a multiple MLS sensor failure. (ordered same as Priorj)
Resbj -- real, mixed, (17,3) (-) (11,3) ; matrix of multiple MLS failure compensated residuals vectors. Cols. are ordered as azim., elev., rng.
refby: ISOLAT, DECIDE

name: NAMES
cont: Character variables which are vectors of sensor names & units
vars: Iyname -- character *9, (13) (6) (7) ; vector of sensor types, Table 6.5
Iyunit -- character *5, (13) (6) (7) ; vector of sensor types, Table 6.5
refby: READFL, DECIDE, HEALR

name: PSIR (not in FINDS2)
cont: Quantities associated with runway yaw
vars: psiru -- real, radians ; runway yaw w.r.t North
simpsr -- real, unitless ; sine of psiru
cospsr -- real, unitless ; cosine of psiru
refby: FINDS/FINDS1, INITXF, SUMIN, SUMOUT, GTOI, RESCMP, HEALR

name: PQRDEG (not in FINDS2)
cont: Computed "best" estimate of P, Q, R (in degrees) as average of all available rate sensors, including standby equipment
vars: apdeg -- real, degrees ; roll rate estimate [(rep1 + rep2)/2]
aqdeg -- real, degrees ; pitch rate estimate
ardeg -- real, degrees ; yaw rate estimate
refby: GTOI, UPDQ

name: RDLOCL
cont: Saved local variables in subroutine READFL. In particular, the saved variables are current sensor measurements to be used at the next iteration and the maximum sensor differences for the data drop-out tests.
vars: Axmold -- real, m/s^2, (2) (-) (2) ; longitudinal accel. previous measurements
Aymold -- real, m/s^2, (2) (-) (2) ; lateral accel. previous measurements

Azmold -- real, m/s^2, (2) (-) (2) ; vertical accel. previous measurements
Pmold -- real, rad/s, (2) (2) (-) ; roll rate gyro previous measurements
Qmold -- real, rad/s, (2) (2) (-) ; pitch rate gyro previous measurements
Rmold -- real, rad/s, (2) (2) (-) ; yaw rate gyro previous measurements
Aziold -- real, rad, (2) (-) (2) ; MLS azimuth previous measurements
Eleold -- real, rad, (2) (-) (2) ; MLS elevation previous measurements
Rngold -- real, m, (2) (-) (2) ; MLS range previous measurments
Airold -- real, m/s, (2) (-) (2) ; IAS previous measurements
Phiold -- real, rad, (2) (2) (-) ; IMU roll previous measurements
Theold -- real, rad, (2) (2) (-) ; IMU pitch previous measurements
Psiold -- real, rad, (2) (2) (-) ; IMU yaw previous measurements
Axmax -- real, m/s^2 ; longitudinal accel. dropout threshold
Aymax -- real, m/s^2 ; lateral accel. dropout threshold
Azmax -- real, m/s ; vertical accel. dropout threshold
Pmax -- real, rad/s ; roll rate gyro dropout threshold
Qmax -- real, rad/s ; pitch rate gyro dropout threshold
Rmax -- real, rad/s ; yaw rate gyro dropout threshold
Azimax -- real, rad ; MLS azimuth dropout threshold
Elemax -- real, rad ; MLS elevation dropout threshold
Rngmax -- real, m ; MLS range dropout threshold
Airmax -- real, m/s ; IAS dropout threshold
Phimax -- real, rad ; IMU roll dropout threshold
Themax -- real, rad ; IMU pitch dropout threshold
Psimax -- real, rad ; IMU yaw dropout threshold

refby: READFL

name: RGOUT (not in FINDS2)
cont: Replicated rate gyro measurements from flight data
vars: Pm -- real, rad/s (2) (2) (-) ; dual roll rate gyro measurements
Qm -- real, rad/s (2) (2) (-) ; dual pitch rate gyro measurements
Rm -- real, rad/s (2) (2) (-) ; dual yaw rate gyro measurements
refby: READFL, SUMIN, GTOI, HEALR

name: SIGTAU (SIG in FINDS1)
cont: Design values for noise parameters used by NFF and detectors
vars: Sig -- real, mixed, (15) (6) (9) s.d. of sensor noise used by NFF (ordered as input sensors, winds, output sensors). Tables D, B
Tau -- real, seconds, (2) (-) (2) ; time constant for horizontal winds in wind model used by NFF
Sigd01 -- real, mixed, (15) (6) (9) ; s.d. of sensor noise for DET01, ordered same as SIG
Sigd05 -- real, mixed, (15) (6) (9) ; s.d. of sensor noise for DET05, ordered same as SIG
Sigd10 -- real, mixed, (15) (6) (9) ; s.d. of sensor noise for DET10, ordered same as SIG
refby: INITG, UPDQ, DECIDE, NOISR

name: <u>SUMLOC</u>
cont: Saved local variables in subroutine SUMIN. In particular, the input sensor measurements from the current iteration are saved to perform trapezoidal integration at the next iteration.
vars: Axmo -- real, m/s^2, (2) (-) (2) ; saved longitudinal accel. measurements
Aymo -- real, m/s^2, (2) (-) (2) ; saved lateral accel. measurements
Azmo -- real, m/s^2, (2) (-) (2) ; saved vertical accel. measurements
Pmo -- real, rad/s, (2) (2) (-) ; saved roll rate gyro measurements
Qmo -- real, rad/s, (2) (2) (-) ; saved pitch rate gyro measurements
Rmo -- real, rad/s, (2) (2) (-) ; saved yaw rate gyro measurements
refby: SUMIN

name: <u>SYNC</u>
cont: Quantities associated with the program timing and synchronization.
vars: dtime -- real, s ; program integration step size (1/20)
idtime -- integer, unitless ; counter incremented at each iteration to compute `time`
time -- real, s ; elapsed time from start of program
tstart -- real, s ; program starting time (default = 0)
tstop -- real, s ; program final time (estimated)
dt22 -- real, s^2 ; saved dtime*dtime/2
idst05 -- real, unitless ; counter to stop/start DET05 after system reconfiguration following failure/healing.
idst10 -- real, unitless ; counter to stop/start DET10 after system reconfiguration following failure/healing.
refby: FINDS/FINDS1/FINDS2, READFL, NAV, INITG, SUMIN, DET01, DET05, DET10, UPDB, UPDQ, DECIDE, HEALR, TLOUT

name: <u>SYSU1</u>
cont: Quantities associated with the inputs to the NFF
vars: nu -- integer, unitless ; total # of inputs to NFF including gravity inputs (default value = 9,3,6)
nul -- integer, unitless ; total # of inputs to NFF associated with an input sensor (i.e, nu -ng), value = 6, 3, 3
nulpl -- integer, unitless ; nul +1 ; = 7, 4, 4
nulc -- integer, unitless ; (nul) - (# of inputs not currently active).
Inoup -- integer, unitless (17) (6) (11) ; pointer vector to absolute input measurements used by NFF (Table 6.3). The array index corresponds to the location in uF1 and the value is the abs. input meas. type index.

Uf1 -- real, mixed, (9) (6) (6) ; vector of compensated inputs used by NFF (computed in SUMIN)

refby: INITG, SUMIN, GTOI, EKFN1, BIASF, BLEND, DET01, DET05, DET10, SETISN, UPDB, UPDH, UPDPH, RESCMP, ISOLAT, DECIDE, RECONF, CLPSIO, NOISR, ADJTBP, HEALR

name: SYSX1

cont: Bias free filter state dimensions and system matrices

vars: nx -- integer, unitless ; total # of states in bias free portion of NFF, value = 11, 3, 8

nx1 - integer, unitless ; nx + 1, value = 12, 4, 9

Af1 -- real, mixed, (11,11) (-) (8,8) ; constant state transition matrix. (Not defined in FINDS1 as it is an identity matrix there).

Bf1 -- real, mixed, (17,9) (3,3) (8,6) ; nonlinear input transition matrix (function of states).

Ef1 -- real, mixed, (17,11) (3,3) (8,8) ; discrete process noise covariance matrix.

refby: INITG, EKFN1, BIASF, BLEND, UPDB, UPDQ, UPDH, UPDPH, ISOLAT, RECONF, CLPSIO, ADJTBP, RCOV

name: SYSXB0

cont: Quantities associated with the bias filter portion of the NFF.

vars: nb -- integer, unitless ; current # of biases estimated by NFF (nb = nub + nyb), value 6, 3, 3

nub -- integer, unitless ; current # of input sensor biases estimated by NFF, value = 6, 3, 3

nyb -- integer, unitless ; current # of measurement biases estimated by NFF, value = 0, 0, 0

nub1 -- integer, unitless ; nub + 1, value = 7, 4, 4

Inobp -- integer, unitless, (13) (6) (7) ; pointer vector to sensor type of each bias estimated. (absolute sensor index) (from Table 6.5)

refby: NAV, INITG, SUMIN, EKFN1, BIASF, BLEND, UPDH, UPDPH, ISOLAT, RECONF, CLPSIO, CLPSBE, ADJTBP

name: SYSYB0 (not in FINDS2)

cont: Variables common to subroutines EKFN1 and BIASF

vars: Rbf0 - real, mixed, (17),12) (3,3) (-) ; saved $HP1*PF1*HP1^T + R$ from EKFN1

Cbf0 -- real, mixed, (17,6) (-) (-) ; bias filter observation matrix.

refby: EKFN1, BIASF

name: SYSW1

cont: Quantities associated with the NFF observation and process noises.

vars: ny -- integer, unitless ; total # of averaged (or collapsed) measurements presented to the NFF, value = 7, 3, 4

nymxi -- integer, unitless ; initial max. # of avgd. meas. to NFF, value = 7, 3, 4

Inoyp -- integer, unitless, (17) (6) (11) ; pointer vector to active avgd. outputs used by NFF. (array index corresponds to the elements of the measurement array & value of each element corresponds to absolute meas. index.) Table 6.2

Inoypi -- integer, unitless, (17) (6) (11) ; inverse mapping of Inoyp, i.e., array index is abs. meas. index and value is the corresponding index in current meas. vector to NFF. If a particular meas. type is not used, its value entry will be zero.

Yf1 -- real, mixed, (7) (3) (4) ; vector of avgd. meas. used by NFF (abs. meas. sensor indexing) Table 6.2

Qf1 -- real, mixed, (8) (3) (5) ; vector of process noise covariances organized by absolute input index, Table 6.4

Hp1 -- real, mixed, (17,17) (3,3) (4,8) ; effective observation matrix for NFF (partial of h w.r.t. x)

Rf1d01 --real, mixed, (7) (3) (4) ; vector of meas. noise covariances used by DET01 (abs. meas. index). Table 6.2

Rf1d05 -- real, mixed, (7) (3) (4) ; vector of meas. noise covariances used by DET05 (abs. meas. index). Table 6.2

Rf1d10 -- real, mixed, (7) (3) (4) ; vector of meas. noise covariances used by DET10 (abs. meas. index).

refby: INITG, SUMOUT, EFKN1, BIASF, BLEND, DET01, DET05, DET10, UPDQ, UPDH, UPDPH, ISOLAT, CLPSIO, NOISR, ADJTBP

name: TRBER

cont: Transformation matrices for various reference frames

vars: Trb -- real, unitless, (3,3) (3,3) (3,3) ; transformation matrix from body axes into the G-frame (for accel. inputs).

	Ter -- real, unitless, (3,3) (3,3) (-) ; matrix relating the body rates to the Euler angles (for gyro inputs).
	Tic -- real, unitless, (3,3) (3,3) (-) ; transformation matrix from runway frame to inertial frame.
refby:	SUMIN, GTOI, UPDB, UPDQ

name:	TSTORE (only in FINDSCMP)
cont:	Temporary scratch areas (matrices) in EKFN1 and BIASF
vars:	Tmp1 -- real, mixed, (17,17) (-) (-) ; local working array
	Tmp2 -- real, mixed, (17,17) (-) (-) ; local working array
refby:	EKFN1, BIASF

name:	UPDQLC
cont:	Saved local variables in subroutine UPDQ
vars:	scalef -- real, unitless ; s.d. of scale factor for rate gyro compensation
	spm -- real, unitless ; average error variance for rate gyro compensation (includes scale factor and misalignment errors)
	dt3 -- real, s^3 ; saved dtime 3/3
refby:	UPDQ

name:	YOBSRV
cont:	Scaling array for the filter observations
vars:	Yscale -- real, mixed, (7) (3) (4) ; vector of scale factors used to scale each avgd. meas. into the NFF. Scaling is performed to ensure that the meas. noise variance is unity for each sensor. (indexed as per Table 6.2)
refby:	INITG, SUMOUT, UPDH, UPDPH, RESCMP, ISOLAT, HEALR

9. REFERENCES

[1] Caglayan, A.K. and Lancraft, R.E., "Fault Tolerant Navigation in a Microwave Landing System Environment," Proc. of the 1982 Position Location and Navigation Symposium, Atlantic City, NJ, December 1982.

[2] Caglayan, A.K., and Lancraft, R.E., "An Aircraft Sensor Fault Tolerant System," NASA CR-165876, April 1982.

[3] Caglayan, A.K., and Lancraft, R.E., "A Fault Tolerant System for an Integrated Avionics Sensor Configuration," NASA CR-3834, September 1984.

[4] Lancraft, R.E. and Caglayan, A.K., "FINDS: A Fault Inferring Nonlinear Detection System, Volume 1: User's Guide," NASA CR-172199, September 1984.

[5] Caglayan, A.K., and Godiwala, P.M., "Evaluation of a Fault Tolerant System for an Integrated Avionics Configuration with TSRV Flight Data," NASA CR-172589, June 1985.

[6] Caglayan, A.K., Godiwala, P.M., and Morrell, F.R., "Performance Analysis of a Fault Inferring Nonlinear Detection System (FINDS) with Integrated Avionics Flight Data," Proceedings of AIAA Computers in Aerospace V Conference, Long Beach, CA., October 1985, AIAA 85-6022.

[7] Caglayan, A.K., and Godiwala, P.M., "A Preliminary Design for Flight Testing the FINDS Algorithm," NASA CR-178043, March 1986.

[8] Caglayan, A.K., Godiwala, P.M., and Morrell, F.R., "Design Considerations for Flight Test of a Fault Inferring Nonlinear Detection System Algorithm for Avionics Sensors," NASA TM-88998, August, 1986

[9] Godiwala, P.M. and Caglayan, A.K., "A Dual-Processor Multi-Frequency Implementation of the FINDS Algorithm," NASA CR-178252, April 1987.

[10] Godiwala, P.M., Caglayan, A.K. and Morrell, F.R., "Evaluation of a Dual Processor Implementation for a Fault Inferring Nonlinear Detection System," AIAA Computers in Aerospace VI Conference, Wakefield, MA, October 1987.

[11] Caglayan, A.K., and Lancraft, R.E., "A Bias Identification and State Estimation Methodology for Nonlinear Systems," Proceedings of the 6th IFAC Symposium on Identification and System Parameter Estimation, Washington D.C., June 1982.

[12] Caglayan, A.K., and Lancraft, R.E., "A Separated Bias Identification and Estimation Algorithm for Nonlinear Systems," Automatica, Vol. 19, No. 5, pp. 561-570, September,1983.

[13] Caglayan, A.K. and Lancraft, R.E., "Reinitialization Issues in Fault Tolerant Systems," Proceedings of the 1983 American Control Conference, San Fransisco, CA, June 1983.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No. NASA CR-178410	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle User's Guide to the Fault Inferring Nonlinear Detection System (FINDS) Computer Program		5. Report Date June 1988
		6. Performing Organization Code
7. Author(s) A.K. Caglayan, P.M. Godiwala, and H.S. Satz		8. Performing Organization Report No. R8801
		10. Work Unit No. 505-66-41-04
9. Performing Organization Name and Address Charles River Analytics Inc. 55 Wheeler Street Cambridge, MA 02138		11. Contract or Grant No. NAS1-17719
		13. Type of Report and Period Covered Contractor Report
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Langley Research Center Hampton, VA 23665-5225		14. Sponsoring Agency Code

15. Supplementary Notes

Langley Technical Monitor: Frederick R. Morrell

16. Abstract

This report describes the operation and internal structure of the computer program FINDS (Fault Inferring Nonlinear Detection System). The FINDS algorithm is designed to provide reliable estimates for aircraft position, velocity, attitude, and horizontal winds to be used for guidance and control laws in the presence of possible failures in the avionics sensors.

The FINDS algorithm was developed with the use of a digital simulation of a commercial transport aircraft and tested with flight recorded data. The algorithm was then modified to meet the size constraints and real-time execution requirements on a flight computer. For the real-time operation, a multi-rate implementation of the FINDS algorithm has been partitioned to execute on a dual parallel processor configuration: one based on the translational dynamics and the other on the rotational kinematics. The report presents an overview of the FINDS algorithm, the implemented equations, the flow charts for the key subprograms, the input and output files, program variable indexing convention, subprogram descriptions, and the common block descriptions used in the program.

17. Key Words (Suggested by Author(s)) sensor failure detection, fault tolerant systems, reliable estimation	18. Distribution Statement Unclassified - Unlimited Subject Category 04

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of pages 90	22. Price A05

NASA FORM 1626 OCT 86

End of Document